Frequency-Domain and Multirate Adaptive Filtering

JOHN J. SHYNK

This paper presents an overview of several frequency-domain adaptive filters that efficiently process discrete-time signals using block and multirate filtering techniques. These algorithms implement a linear convolution that is equivalent to a block time-domain adaptive filter, or they generate a circular convolution that is an approximation. Both approaches exploit the computational advantages of the FFT. Subband adaptive filtering is also briefly described whereby the input data are first processed by a bank of narrow-band bandpass filters that are approximately nonoverlapping. The transformed signals are then decimated by a factor that depends on the degree of aliasing that can be tolerated, resulting in a large computational savings. Several performance issues are considered, including convergence properties and computational complexities of the adaptive algorithms, as well as the effects of circular convolution and aliasing on the converged filter coefficients.

1053-588/92/$3.00 © 1992 IEEE

In recent years, there has been widespread interest in adaptive signal processing applications that require filters with very long impulse responses [1]. It is not uncommon, for example, that thousands of FIR (finite impulse response) filter coefficients are needed to achieve the desired level of performance in channel equalization [2]-[4], adaptive noise cancellation [5], [6], acoustic echo cancellation [7]-[12], as well as numerous other applications (see, e.g., [13]-[16]). One solution to this complexity problem has been to use adaptive IIR (infinite impulse response) filters, such that an effectively long impulse response can be achieved with relatively few filter coefficients. The complexity advantages of adaptive IIR filters are well known, but they are not yet widely used because of potential problems with algorithm instability, slow convergence, and local minima [17]-[19].

An alternative approach to reducing the computational complexity of large adaptive FIR filters is to incorporate *block* updating strategies whereby FFT (fast Fourier transform) algorithms [20] efficiently perform the filter convolution and the gradient correlation [21]-[28]. These techniques reduce the complexity because the filter output and the adaptive weights are computed only after a large block of data has been accumulated. In addition to block convolution methods, computationally efficient adaptive algorithms based on *subband* techniques have recently been developed [7], [29], [6], [30]. Both of these types of adaptive filters can be viewed as multirate signal processors [31], and it is their capability to perform certain functions at a lower sampling rate that allows for a reduction in the complexity. Depending on how the data is organized, these approaches may introduce some degradation in performance, including an end-to-end *delay* and possibly a reduction in the stable range of the algorithm step size [26]. For nonstationary signals, the *tracking* performance of a block algorithm also generally becomes worse [32].

The basic operation underlying a frequency-domain adaptive filter is the transformation of the input signal into a more desirable form before the adaptive processing. This is accomplished by one or more discrete Fourier transforms (DFTs) or filter banks whereby the input signal is transformed to the frequency domain as shown in Fig. 1. The transformation is nonadaptive and corresponds to a simple preprocessing step that is *independent* of the data. Observe that two formulations are shown:[1] (a) the error $e(n)$ is computed in the time domain which is then transformed, or (b) the desired response $d(n)$ is first transformed and the error is computed directly in the frequency domain. For adaptive algorithms where the error is a linear function of the data (e.g., the least-mean-square (LMS) algorithm [33]), these two approaches may yield similar results. However, for algorithms that have nonlinear error functions (e.g., the constant modulus algorithm (CMA) [34]), the two structures can lead to very different results and only one may provide acceptable performance. The configuration that performs best will also depend on the type of filter banks (or transforms) and the proposed application.

Frequency-domain adaptive filters have primarily two advantages compared to time-domain implementations [26], [27]. The first advantage is the potentially large savings in the computational complexity, as previously mentioned. The FFT is an efficient implementation of the DFT which provides this savings. A second advantage is that the DFT and the filter bank structures generate signals that are approximately *uncorrelated* (orthogonal) [35]. As a result, a time-varying step size can be used for each adaptive weight, thereby allowing a more uniform convergence rate across the adaptive filter. It is well known that the eigenvalue disparity of the input signal correlation matrix generally determines the convergence rate of a gradient-descent algorithm [33]. These eigenvalues correspond roughly to the power of the signal spectrum at equally-spaced frequency points around the unit circle [36]-[38]. It is therefore possible to compensate for this power variation by using step sizes that are inversely proportional to the power levels in the DFT frequency bins.[2] As a result, the overall convergence rate of the algorithm may be improved, sometimes approaching that achievable with RLS (recursive-least-squares) algorithms [39] without a similar increase in the computational complexity.

The goal of this paper is to provide a common framework for several frequency-domain and subband adaptive filters that are available today. Two kinds of implementations are considered. The first one involves a straightforward efficient realization of block time-domain adaptive algorithms using FFT filtering techniques. This approach yields either a linear convolution or a circular convolution depending on how the data samples are organized. The second implementation is based on filter-bank techniques and subband filtering whereby the internal signals are downsampled (decimated[3]) before the adaptive processing. If the filter banks are chosen appropriately, the distorting effects of aliasing can be sufficiently controlled. Both of these implementations will be referred to as frequency-domain adaptive filters (FDAFs) even though they may not always utilize the DFT.

Although the focus of this overview is on FIR filters and the LMS algorithm, there is a brief discussion of frequency-domain adaptive IIR filtering for both the equation-error and output-error formulations [19]. Nonlinear error functions are also considered, such as those encountered with CMA and decision-directed equalizers [40]. All of the configurations discussed here can be modified to handle a variety of adaptive algorithms, and their complexity can often be reduced

Photo Credit: FPG, M. Simpson

[1] Hybrids of these two basic formulations are also possible.

[2] Frequency bin m corresponds to the band of frequencies centered at $\omega_m = e^{j2\pi m/N}$, $m = 0, \ldots, N-1$, (where $j = \sqrt{-1}$) with a bandwidth of approximately $2\pi/N$. A "frequency bin" can be similarly defined for other types of filter banks.

[3] Decimation actually refers to downsampling by a factor of 10% (i.e., every tenth sample is discarded), but it is generally used to denote downsampling by any factor [31].

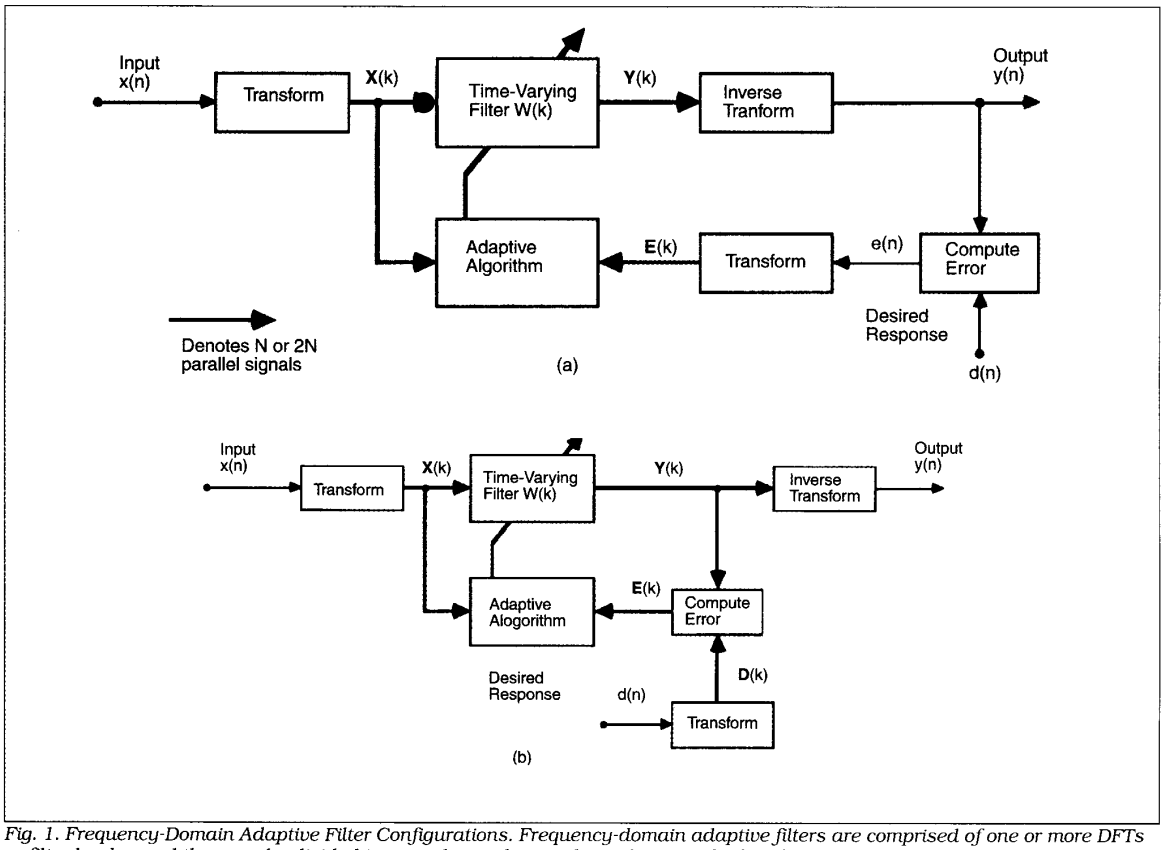


Fig. 1. Frequency-Domain Adaptive Filter Configurations. Frequency-domain adaptive filters are comprised of one or more DFTs or filter banks, and they can be divided into two basic classes depending on whether the error signal is computed in (a) the time domain or (b) the frequency domain. The time-varying weight vector, **W** *(k), usually corresponds to an FIR filter, although there are some IIR implementations. The adaptive algorithm is a gradient-descent method, and a* **time-varying** *step size is often included to improve the convergence rate.*

further by simplifying certain components, depending on the performance requirements of the application.

Throughout the paper, the following notation will be used. Time-domain variables are denoted by lower-case letters whereas frequency-domain variables are assigned upper-case letters. Vectors and matrices will be given in bold font while scalars are written in italic font.

BLOCK ADAPTIVE FILTERING

The standard time-domain LMS algorithm has the following nonblock coefficient (weight) update [41], [33]:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + 2\mu\boldsymbol{x}(n)e(n) \quad (1)$$

where $\mathbf{w}(n)$ is the adjustable weight vector and $\mathbf{x}(n)$ is the input signal vector defined as

$$\boldsymbol{w}(n) = [w_0(n),\ldots,w_{N-1}(n)]^T \quad (2)$$

$$\boldsymbol{x}(n) = [x(n),\ldots,x(n-N+1)]^T \quad (3)$$

The number of coefficients is N and the superscript T denotes matrix/vector transpose. The positive scalar step size μ controls the convergence rate and steady-state performance of the algorithm. The output error is given by $e(n) = d(n) - y(n)$, the filter output is obtained as the inner product

$$y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n) \quad (4)$$

and $d(n)$ is the desired response signal which depends on the application of the adaptive filter. The gradient *estimate*, $\hat{\nabla}(n)$, is simply the derivative of $e^2(n)$ with respect to $\mathbf{w}(n)$ and it is given by[4]

$$\hat{\nabla}(n) = \boldsymbol{x}(n)e(n) \quad (5)$$

The LMS algorithm attempts to minimize the mean-square error (MSE) performance function, defined as $\xi = E[e^2(n)]$ where $E[\cdot]$ is the expectation operator. It is based on the method of steepest descent [42] and its convergence properties are well known for a variety of signal conditions.

The recursion in (1) is done each time a new sample pair $\{x(n), d(n)\}$ is received. Alternatively, one could keep the weights *fixed* until N data pairs are received and then incorporate this information to update the weights

[4] The gradient is usually defined with a minus sign, but for convenience we will use the definition in (5).

only once during this period. Consider the following weight update:

$$\mathbf{w}(n+L) = \mathbf{w}(n+L-1) + 2\mu\mathbf{x}(n+L-1)e(n+L-1) \quad (6)$$

which is similar to (1) except for the incremented time arguments where $1 \le L \le N$ is an integer. By substituting the recursion for $\mathbf{w}(n + L - 1)$ in terms of $\mathbf{w}(n + L - 2)$ and continuing this substitution until we reach $\mathbf{w}(n)$, it is straightforward to show that[5]

$$\mathbf{w}(n+L) = \mathbf{w}(n) + 2\mu\sum_{m=0}^{L-1}\mathbf{x}(n+m)e(n+m) \quad (7)$$

This expression represents a single update of the weights from time n to time $n+L$ based on the L data samples accumulated, and it is thus called a *block* recursion. It is important to note that the error terms in the summation all depend on the *same* weight vector $\mathbf{w}(n)$, i.e., $e(n+m) = d(n+m) - y(n+m)$, $m = 0, \ldots, L-1$, where

$$y(n+m) = \mathbf{x}^T(n+m)\mathbf{w}(n) \quad (8)$$

Consequently, (7) differs from the standard nonblock LMS algorithm where each new error depends on the most recent update of the weights. In this paper, we are interested primarily in the case of $L = N$ because this is the most efficient value for the FFT algorithms [24]. However, for the sliding DFT [43] and frequency-sampling [44] structures briefly described later, $L = 1$, and the adaptive algorithm operates at the incoming data rate corresponding to the nonblock update in (1). These adaptive filters do not have the computational advantages of the multirate adaptive filters, but they do exploit the orthogonality properties of the DFT, and they suggest how to derive the more efficient subband implementations.

Since (7) is a block update that operates at a lower sampling rate than that of the incoming data, it will be convenient to define a new time index k where one increment corresponds to L increments of the original index n. Without loss of generality, we can substitute $n = kL$ where n is an integer multiple of k. By factoring the argument $kL+L$ on the left-hand side of (7) as $(k+1)L$ and dropping the explicit dependence of the weight vector on L, we have the following equivalent block update:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\sum_{m=0}^{L-1}\mathbf{x}(kL+m)e(kL+m) \quad (9)$$

Thus, k refers to block time and n denotes the original time index of the incoming data. The block gradient estimate is given by the following summation:

$$\hat{\nabla}(k) = \sum_{m=0}^{L-1}\mathbf{x}(kL+m)e(kL+m) \quad (10)$$

[5]The arguments of $\mathbf{x}(n+m)$ and $e(n+m)$ indicate that "future" data samples are to be used in the block update. However, (7) should not be interpreted as being a noncausal operation.

and it is derived by differentiating the block MSE, defined as $\xi_L = E[\mathbf{e}^T(k)\mathbf{e}(k)]$ where $\mathbf{e}(k) = [e(kL), \ldots, e(kL+L-1)]^T$. For wide-sense stationary signals, it can be shown that $\xi_L = L\xi$.

The block LMS (BLMS) algorithm in (9) essentially minimizes the same MSE performance function as the nonblock LMS algorithm in (1). One can view the block gradient in (10) as a more accurate estimate of the true (ensemble) gradient because L terms are being averaged for each update (see Fig. 2). For wide-sense stationary signals, the steady-state weight vector (Wiener solution), misadjustment, and time constants of the BLMS algorithm are identical to those of the standard LMS algorithm. The main difference is that the maximum value of the step size such that the algorithm is stable

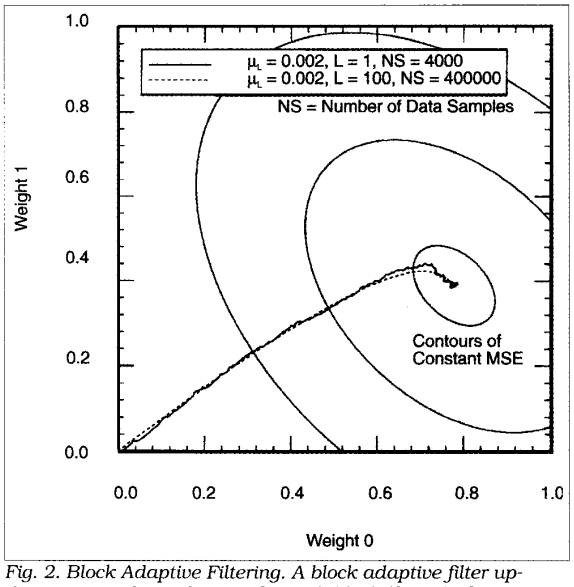


Fig. 2. Block Adaptive Filtering. A block adaptive filter updates its weights only once for each block (frame) of new data, where $L \ge 1$ is the block size. In effect, the gradient estimate is computed as an **average** *of the data, and therefore it is a more accurate representation of the ensemble gradient vector. At each step of the algorithm, the descent direction is more closely aligned with the true gradient, and smoother convergence is possible because the estimate is less "noisy." The block update in (9) can be rewritten as*

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu L\left(\frac{1}{L}\sum_{m=0}^{L-1}\mathbf{x}(kN+m)\,e(kN+m)\right)$$

$$= \mathbf{w}(k) + 2\mu_L\hat{\nabla}_L(k)$$

where $\mu_L = \mu L$ is the effective step size and $\hat{\nabla}_L(k)$ is the averaged gradient estimate. It is clear from this form of the update that the stability bound for μ and the variance of $\hat{\nabla}_L(k)$ are both inversely proportional to L. The averaged gradient estimate thus becomes more accurate as the block size increases. This improvement does not imply faster convergence, however, because the eigenvalue spread of the input signal correlation matrix remains unchanged for any value of L. In order to have a similar convergence time as the nonblock algorithm using the **same** *value of μ_L, the block LMS algorithm requires more data (by a factor of L) because of the gradient averaging.*

is now scaled down by a factor of L [26], [27], [45]. If the input signal correlation matrix has a large eigenvalue spread, then the BLMS algorithm may converge more slowly than the LMS algorithm because of the tighter upper bound on μ. This could be a problem in applications which require fast convergence and need a large value for the step size.

Observe that the block gradient in (10) is a *linear correlation* between the error signal and the input signal vector, and that (8) is a *linear convolution* between the weights and the input signal vector. It is possible, therefore, to efficiently implement each of these sums by taking discrete Fourier transforms, computing their product, and then inverse transforming the result [46]. Thus, we see that there are tradeoffs with the block algorithm. It has smoother convergence properties and can be efficiently realized using FFTs, but the maximum achievable rate of convergence is reduced because the maximum value of the step size is necessarily smaller. The block updating can also impair the tracking performance of the algorithm, especially if the block size is large and the data are highly nonstationary.

GENERAL FORM OF THE FDAF ALGORITHMS

The frequency-domain adaptive algorithms described in this paper all have a recursion that is similar to the block update in (9). Because DFT computations inherently perform a circular convolution [46], the adaptive filters generally require data *constraints* in order to implement the desired linear convolution. These constraints force certain elements of the signal vectors to be zero and only a subset of the components are retained for later use in the algorithm. Generally, by removing one or more of these constraints the FDAF will have a reduced computational complexity, but there may be a degradation in performance because of the wrap-around effects introduced by circular convolutions [47], [26]. The algorithm may not converge to the desired Wiener solution.

Analogous to (2) and (3), define the frequency-domain weight vector as

$$\boldsymbol{W}(k) = [W_0(k),\ldots,W_{M-1}(k)]^T \tag{11}$$

and the input signal *matrix* as

$$\boldsymbol{X}(k) = \text{diag}\{X_0(k),\ldots,X_{M-1}(k)\} \tag{12}$$

where diag$\{\cdot\}$ is an operator that forms a diagonal matrix. The number of elements, M, depends on the FDAF configuration (usually $M = N$ or $M = 2N$); this will be discussed separately later for each case. Although we could represent $\mathbf{X}(k)$ as a vector, it will be more convenient to use this matrix definition so that the frequency-domain output vector can be written as the following matrix/vector product:

$$\boldsymbol{Y}(k) = \boldsymbol{X}(k)\boldsymbol{W}(k) \tag{13}$$

The components of $\mathbf{Y}(k)$ are defined in a manner similar to $\mathbf{W}(k)$ in (11).

With these definitions, the general form of the FDAF algorithms can be expressed as

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mathbf{G}\boldsymbol{\mu}(k)\boldsymbol{X}^H(k)\boldsymbol{E}(k) \tag{14}$$

where the superscript H denotes complex conjugate transpose. The frequency-domain weight vector is generally complex-valued so that (14) is a version of the complex LMS algorithm [48]. The time-varying matrix $\boldsymbol{\mu}(k)$ is diagonal and it contains the step sizes $\mu_m(k)$, $m = 0, \ldots, M-1$. Generally, each step size is varied according to the signal power in that frequency bin. For example, we could have $\mu_m(k) = \mu / P_m(k)$ where μ is a fixed scalar and $P_m(k)$ is an estimate of the signal power in the m^{th} bin, which might be computed as (see, e.g., [49], [27], [38])

$$P_m(k) = \lambda P_m(k-1) + \alpha |X_m(k)|^2 \tag{15}$$

where $\lambda = 1 - \alpha$ is a forgetting factor.[6] If the data are statistically stationary, the step-size matrix may be fixed such that $\mu_m(k) = \mu_m$, or each step size may even be identical, i.e., $\boldsymbol{\mu}(k) = \mu\boldsymbol{I}$ where $\mathbf{I}$ is the identity matrix. The initial value, $P_m(0)$, is chosen to be a positive number, δ_m, which depends on some initial power estimate of $X_m(k)$. As another example, the step size could be computed simply as $\mu_m(k)=\mu/(\alpha+|X_m(k)|^2)$ where α is a small positive constant that bounds $\mu_m(k)$ when $X_m(k)$ is momentarily small. The corresponding algorithm is similar to the normalized LMS algorithm [50].

The error vector, $\mathbf{E}(k)$, is defined analogous to $\mathbf{W}(k)$ and its components depend on the specific FDAF algorithm. In some cases, the error is computed in the time domain and $\mathbf{E}(k)$ is simply its frequency-domain transformation. In other cases, the desired response is first transformed to $\mathbf{D}(k) = [D_0(k), \ldots, D_{M-1}(k)]^T$ and the error is computed directly in the frequency domain as a function of $\mathbf{Y}(k)$ and $\mathbf{D}(k)$. The matrix $\mathbf{G}$ represents a constraint on the gradient, $\mathbf{X}^H(k)\,\mathbf{E}(k)$, which must be imposed in order to achieve a linear correlation. This constraint is usually defined as a restriction on the time-domain signals, and $\mathbf{G}$ is its frequency-domain transformation. Observe that $\mathbf{G}$ *premultiplies* $\boldsymbol{\mu}(k)$ in (14), indicating that the constraint also applies to the step-size matrix. If $\boldsymbol{\mu}(k) = \mu\boldsymbol{I}$, then μ can be factored out to the left of $\mathbf{G}$; for convenience, this simplified form will be used later for each of the FDAF algorithms, but it is clear that we can always incorporate a time-varying step-size matrix.

In order to simplify the notation, it will be convenient to define $\mathbf{F}$ as the $M \times M$ DFT matrix with elements $F_{ml} = e^{-j2\pi\, ml\,/M}$. Its inverse is given by $\mathbf{F}^{-1} = \mathbf{F}^H / M$ such that $\mathbf{F}^H\,\mathbf{F} = M\,\mathbf{I}_M$ where the subscript M denotes the dimension of the identity matrix. Thus, premultiplying a vector by $\mathbf{F}$ will compute the DFT of its components, and the inverse DFT (IDFT) matrix $\mathbf{F}^{-1}$ will similarly

[6] The forgetting factor ($0 < \lambda < 1$) controls the effective memory, $\tau = 1/(1-\lambda)$, of the power estimate.

compute the time-domain samples. A subscript will not be included on $\mathbf{F}$ since its size will be clear from the particular FDAF algorithm.

CONVOLUTION IMPLEMENTATIONS

Linear and Circular Convolution

As mentioned before, the product of two DFT sequences yields a circular convolution (or a circular correlation depending on how the sequence elements are ordered). A circular convolution can differ markedly from a linear convolution such that the resulting sequence appears to be the output of a periodically time-varying filter [47]. Clearly, this is not acceptable if the desired result is a linear convolution, such as the output signal computed in (8). Fortunately, it can be shown that certain elements of the circular convolution correspond to a subset of the linear convolution [46], the size of which depends on the relative lengths of the two sequences, as shown in Fig. 3.

There are two well-known techniques for performing a linear convolution using FFT algorithms, and these are referred to as the *overlap-save* and *overlap-add* sectioning methods [51]. By overlapping elements of the data sequences and retaining only a subset of the final DFT product, a linear convolution between a finite-length sequence and an infinite-length sequence is readily obtained. In our case, the frequency-domain weight vector corresponds to the finite-length "sequence," and the input signal corresponds to the infinite-length sequence. In order to generate N correct output samples, it will be necessary to use DFTs of length $\geq 2N\text{-}1$. It turns out that $2N$-point DFTs ($M = 2N$) are suitable for our purposes and that the optimal block size is $L = N$ [26].

Overlap-Save Method

Consider first the process of computing the filter output in (8). Let $\mathbf{W}(k)$ and $\mathbf{X}(k)$ be derived from the corresponding time-domain quantities as

$$\boldsymbol{W}(k) = \boldsymbol{F}[\boldsymbol{w}^T(k), 0,\ldots,0]^T \tag{16}$$

and

$$\boldsymbol{X}(k) = \text{diag}\{\boldsymbol{F}[x(kN-N), \ldots,x(kN-1), x(kN),\ldots,x(kN+N-1)]^T\} \tag{17}$$

where $\mathbf{w}(k)$ is defined in (2) with n replaced by k. According to the overlap-save method, N output samples $\mathbf{y}(k) = [y(kN), \ldots, y(kN+N\text{-}1)]^T$ from a linear convolution can be computed as

$$\boldsymbol{y}(k) = \text{last } N \text{ components of } \boldsymbol{F}^{-1}\mathbf{Y}(k) \tag{18}$$

where $\mathbf{Y}(k)$ is the frequency-domain output vector in (13). The input sequence in (17) contains N samples

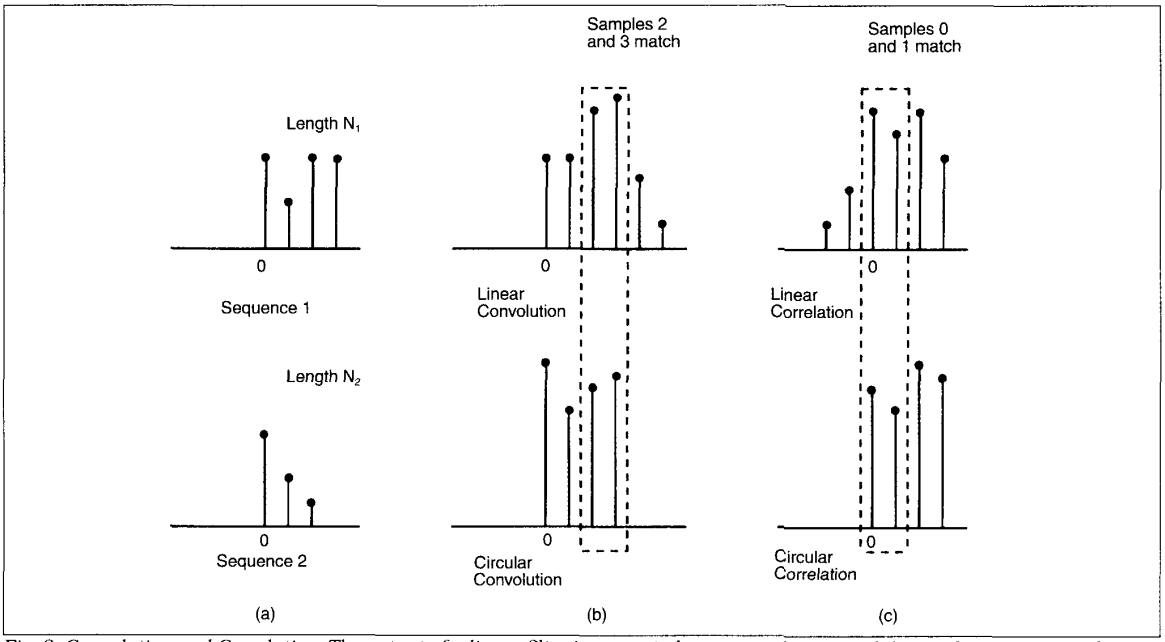


Fig. 3. Convolution and Correlation. The output of a linear filter is computed as a convolution, and the gradient estimate of an adaptive algorithm is a correlation. The convolution of two sequences requires that one sequence be reversed before the samples are shifted, multiplied, and added together. Correlation is similar to convolution, except that neither sequence is reversed. It can be shown that a **subset** *of the samples from a circular convolution (correlation) is identical to a specific set of samples of a linear convolution (correlation). If the two sequences in (a) have lengths N_1 and N_2 where $N_1 \geq N_2$, then the* **last** *$N_1\text{-}N_2\text{+}1$ samples of a circular convolution correspond to a linear convolution, as shown in (b). On the other hand, the* **first** *$N_1\text{-}N_2\text{+}1$ samples of a circular correlation correspond to a linear correlation, as shown in (c). Thus, a linear convolution or correlation can be generated from the appropriate circular one by first* **sectioning** *and* **overlapping** *the data and then retaining a subset of the final result.*

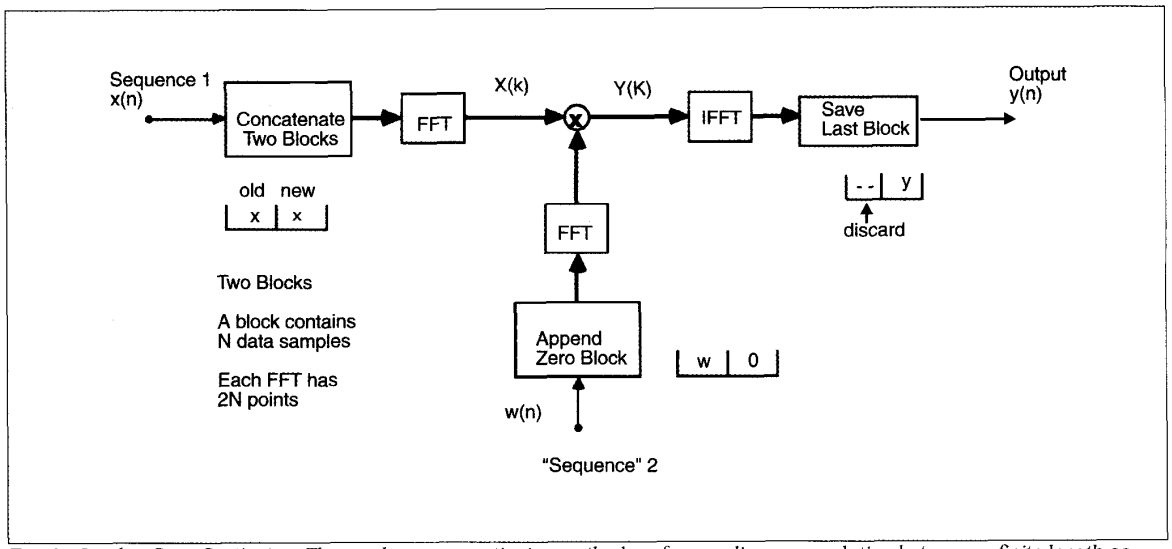


Fig. 4. Overlap-Save Sectioning. The overlap-save sectioning method performs a linear convolution between a finite-length sequence and an infinite-length sequence by appropriately partitioning the data. The finite-length "sequence" $\mathbf{w}(n)$ (in our case, the adaptive weights) has N elements; after appending N zeros, a 2N-point FFT is computed. For the infinite-length input sequence $\mathbf{x}(n)$, the most recent N data samples are concatenated with the previous block of N samples; a 2N-point DFT of this extended data vector is then computed. The product of the transformed sequences (i.e., $\mathbf{Y}(k) = \mathbf{X}(k)\,\mathbf{W}(k)$) is processed by a 2N-point inverse FFT (IFFT), yielding a block of output samples. The first N points of this output frame are discarded, while the last N points are the desired output samples of a linear convolution.

from the current block of data and another N samples from the *previous* block; in effect, the data are being overlapped by N points so that only N new samples are introduced before the DFT is computed for each block update (since the DFT size is $2N$, this is referred to as 50% overlap). Only the last N points of the IDFT of $\mathbf{Y}(k) = \mathbf{X}(k)\,\mathbf{W}(k)$ are retained because the first N terms correspond to a circular convolution. Figure 4 illustrates how the data are partitioned and shows a block-diagram representing (13) and (16) - (18). These equations are entirely equivalent to computing the inner product in (8) N times, but it requires less complexity because of the efficiency of the FFT. This is a standard technique that is frequently used for non-adaptive filtering where the weight vector is time invariant.

A similar technique can be employed to implement the block adaptive algorithm because the gradient in (10) is a linear correlation and the weights are fixed for the entire block of N samples. The error terms are computed in the time domain according to $e(kN+m) = d(kN+m) - y(kN+m)$, $m = 0, \ldots, N-1$, and this block, grouped as $\mathbf{e}(k) = [e(kN), \ldots, e(kN+N-1)]^T = \mathbf{d}(k) - \mathbf{y}(k)$ where $\mathbf{d}(k) = [\,d(kN), \ldots, d(kN+N-1)\,]^T$, is transformed to the frequency domain as follows:

$$\mathbf{E}(k) = \mathbf{F}[0,\ldots,0,\ \mathbf{e}^T(k)]^T \qquad (19)$$

The error vector is augmented with N zeros because N terms of the output are discarded to implement the linear convolution in (18). Alternatively, one may view $\mathbf{e}(k)$ as having the same role in the correlation as $\mathbf{w}(k)$ does in the convolution, except that the zeros *precede* $\mathbf{e}(k)$ because a correlation is basically a "reversed" convolution (see Fig. 3). Applying the same reasoning as was used to derive the block output, it is straightforward to show that the block gradient estimate is

$$\hat{\nabla}(k) = \text{first } N \text{ components of } \mathbf{F}^{-1}\mathbf{X}^H(k)\mathbf{E}(k) \qquad (20)$$

where the first N elements are retained (as opposed to the last N elements in (18) — again, because the gradient is a correlation). This expression contains N terms that correspond exactly to the time-domain block gradient defined in (10).

The final step of the algorithm transforms this time-domain gradient into its frequency-domain counterpart, which is then added to $\mathbf{W}(k)$ in order to generate the updated weights $\mathbf{W}(k+1)$. Because $\mathbf{w}(k)$ is followed by N zeros in (16), the gradient in (20) must be similarly augmented. The algorithm is thus given by

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\mu\mathbf{F}[\hat{\nabla}^T(k),\ 0,\ldots,0]^T \qquad (21)$$

which is equivalent to the update in (9) except that DFTs have been used to implement the output convolution and the gradient correlation. The complete FDAF structure is shown in Fig. 5 where we see that a total of five DFTs are needed to realize the entire adaptive filter. It is important to note that (21) is not an exact realization of the sample-by-sample time-domain LMS algorithm in (1) because the weights here are kept fixed for every block of N samples while the data are accumulated. This implementation was independently derived by Clark et al. [22], [52] and Ferrara [23].

The last two DFTs and the associated data sectioning are labeled in Fig. 5 as a *gradient constraint*. Because there are only N weights in the time domain, there should be an "equivalent" number in the frequency domain; i.e., if we inverse transform the $2N$ frequency-

domain weights to the time domain via an IDFT, only the first N transformed weights should be nonzero. The frequency-domain weight vector in (16) is generated by appending N zeros to the actual weights, but the last N terms of $\mathbf{F}^{-1}\,\mathbf{X}^{H}(k)\,\mathbf{E}(k)$ are usually nonzero. The gradient constraint ensures that the correct weight update is performed. Note also that the initial weight vector, $\mathbf{W}(0)$, must be chosen such that the last N terms of its IDFT are zero.

Because the overlap-save FDAF is simply an efficient implementation of the BLMS algorithm, it has the same convergence properties in terms of misadjustment, convergence speed, and the stable range of the step size μ. Furthermore, the adaptive weights converge to the same Wiener weight vector, yielding the same steady-state (minimum) MSE. If a different step size is used for each adaptive weight, the convergence rate of the algorithm can be improved without increasing this minimum MSE

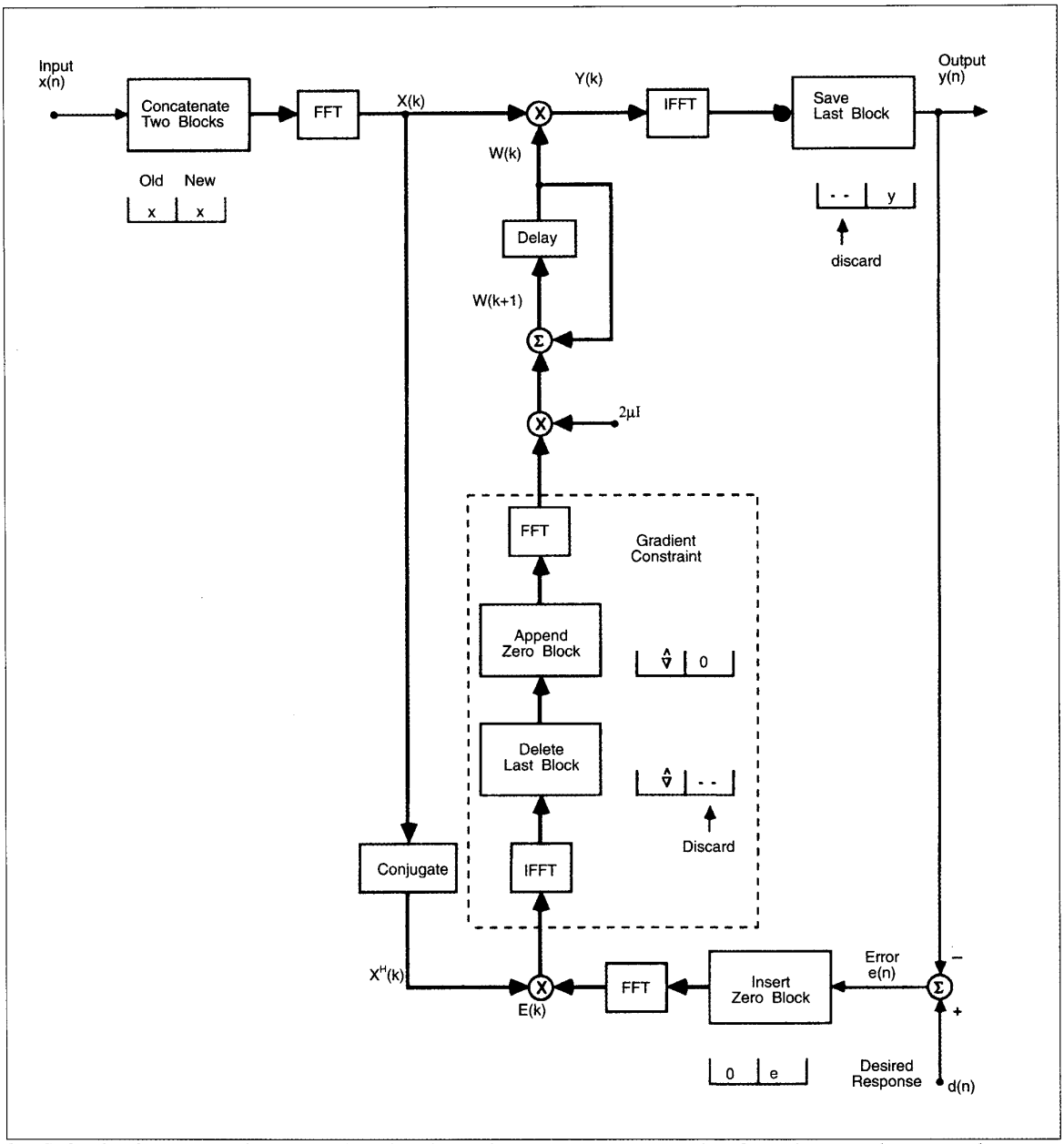


Fig. 5. Overlap-Save FDAF. This FDAF is based on the overlap-save sectioning procedure for implementing linear convolutions and linear correlations. The gradient constraint ensures that the IDFT of the 2N frequency-domain weights yields only N non-zero time-domain weights. Because the DFTs are computed only once for each block of data, there is an **end-to-end** *delay of N samples.*

[38], [53]. Thus, the overlap-save FDAF not only has a reduced computational complexity, but its convergence rate can also be improved by compensating for the signal power variation across the frequency bins.

Algorithm Constraints and a Matrix Formulation

It is convenient to rewrite the operations of the overlap-save FDAF using a matrix formulation so that the algorithm can be represented by the general form in (14). For the operation described by (20) where only the first N components of the IDFT are retained, we can define a time-domain constraint matrix $\mathbf{g}$ as follows:[7]

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{I}_N & \boldsymbol{0}_N \\ \boldsymbol{0}_N & \boldsymbol{0}_N \end{bmatrix} \tag{22}$$

where $\mathbf{0}_N$ is an $N \times N$ matrix of zeros. Using this $2N \times 2N$ constraint matrix, the gradient in (20) can be rewritten more compactly as

$$[\hat{\nabla}^{\boldsymbol{T}}(k),\, 0,\ldots,0]^T = \boldsymbol{g}\boldsymbol{F}^{-1}\boldsymbol{X}^H(k)\boldsymbol{E}(k) \tag{23}$$

so that the weight update in (21) becomes

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mu\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\boldsymbol{X}^H(k)\boldsymbol{E}(k) \tag{24}$$

Comparing this expression with the general form in (14), we see that the gradient constraint is simply $\mathbf{G} = \mathbf{F}\,\mathbf{g}\,\mathbf{F}^{-1}$, which is a full matrix (but with rank $\leq 2N$). It is also possible to rewrite the output vector in (18) using a similar matrix representation:

$$\boldsymbol{y}(k) = \boldsymbol{k}\boldsymbol{F}^{-1}\boldsymbol{Y}(k) \tag{25}$$

where $\mathbf{k}$ is the following $N \times 2N$ constraint matrix:

$$\boldsymbol{k} = [\, \boldsymbol{0}_N \quad \boldsymbol{I}_N \,] \tag{26}$$

Although $\mathbf{g}$ and $\mathbf{k}$ simplify the description of the algorithm and provide a convenient representation for its analysis, one would not directly compute $\mathbf{G} = \mathbf{F}\,\mathbf{g}\,\mathbf{F}^{-1}$ in an actual implementation; FFT algorithms should be used instead. For convenience, the complete algorithm employing this matrix notation is summarized in Table I. Observe that the error vector has also been rewritten using $\mathbf{k}$ as

$$\boldsymbol{E}(k) = \boldsymbol{F}\boldsymbol{k}^T\boldsymbol{e}(k) \tag{27}$$

All other quantities in the table have been previously defined.

Consider the following time-domain constraint matrix which is similar to $\mathbf{g}$ in (22):

$$\tilde{\boldsymbol{g}} = \begin{bmatrix} \boldsymbol{0}_N & \boldsymbol{0}_N \\ \boldsymbol{0}_N & \boldsymbol{I}_N \end{bmatrix} \tag{28}$$

Instead of using the constraint $\mathbf{k}$ to generate $\mathbf{y}(k)$, we can write (18) as

[7] This matrix is also called a window function [38].

TABLE I.
FDAF ALGORITHM BASED ON OVERLAP-SAVE SECTIONING

INITIALIZATION:

$$\boldsymbol{W}(0) = [0,\ldots,0]^T$$

$$P_m(0) = \delta_m, \quad m = 0,\ldots,2N-1$$

MATRIX DEFINITIONS:

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{I}_N & \boldsymbol{0}_N \\ \boldsymbol{0}_N & \boldsymbol{0}_N \end{bmatrix}, \quad \boldsymbol{k} = [\, \boldsymbol{0}_N \;\; \boldsymbol{I}_N \,]\text{; sectioning constraints}$$

$$\boldsymbol{F} = 2N \times 2N \text{ DFT matrix}$$

FOR EACH NEW BLOCK OF N INPUT SAMPLES:

$$\boldsymbol{X}(k) = \text{diag}\{\boldsymbol{F}[x(kN-N),\ldots,x(kN-1),\, x(kN),\ldots,x(kN+N-1)]^T\}$$

$$\boldsymbol{Y}(k) = \boldsymbol{X}(k)\boldsymbol{W}(k)$$

$$\boldsymbol{y}(k) = \boldsymbol{k}\boldsymbol{F}^{-1}\boldsymbol{Y}(k)$$

$$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{y}(k)$$

$$\boldsymbol{E}(k) = \boldsymbol{F}\boldsymbol{k}^T\boldsymbol{e}(k)$$

$$P_m(k) = \lambda P_m(k-1) + \alpha\,|X_m(k)|^2, \quad m = 0,\ldots,2N-1$$

$$\mu(k) = \mu\ \text{diag}\{P_0^{-1}(k),\ldots,P_{2N-1}^{-1}(k)\}$$

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\,\boldsymbol{\mu}\,(k)\boldsymbol{X}^H(k)\boldsymbol{E}(k)$$

$$[0,\ldots,0,\, \boldsymbol{y}^T(k)]^T = \tilde{\boldsymbol{g}}\boldsymbol{F}^{-1}\boldsymbol{Y}(k) \tag{29}$$

which is augmented with zeros and has a form similar to the gradient estimate in (23). Define the *constrained* frequency-domain output as $\mathbf{Y}'(k) = \mathbf{F}[\,0, \ldots, 0\,,\, \mathbf{y}^T(k)\,]^T$. From (29) we see that $\mathbf{Y}'(k) = \mathbf{F}\tilde{\mathbf{g}}\,\mathbf{F}^{-1}\,\mathbf{Y}(k)$ is the relationship between the constrained and unconstrained frequency-domain output signal vectors. The equivalent frequency-domain constraint is $\tilde{\mathbf{G}} = \mathbf{F}\,\tilde{\mathbf{g}}\,\mathbf{F}^{-1}$ (which is similar to $\mathbf{G} = \mathbf{F}\,\mathbf{g}\,\mathbf{F}^{-1}$). Next, define the frequency-domain desired response vector as $\mathbf{D}(k) = \mathbf{F}\,[\,0\,,\ldots,\,0\,,\,\mathbf{d}^T(k)\,]^T$, which is analogous to the error vector in (19). Thus, it is clear that the frequency-domain error could also be computed according to $\mathbf{E}(k) = \mathbf{D}(k) - \mathbf{Y}'(k)$. Of course, this representation would not actually be used to implement the algorithm because an additional DFT is required, but it emphasizes the fact that $\mathbf{E}(k)$ is determined by the constrained output $\mathbf{Y}'(k)$ (i.e., $\mathbf{E}(k)$ cannot be derived simply as the difference $\mathbf{D}(k) - \mathbf{Y}(k)$). This result illustrates that either configuration in Fig. 1 can be used to implement the overlap-save FDAF, but it is more efficient to compute the error in the time domain.

From the general form of the algorithm, it is obvious that other types of gradient constraints are possible and these may lead to improved performance in certain

applications. For these other constraints, the algorithm may not be an exact implementation of the block adaptive filter in (9). For example, it has been shown in [38] that the convergence rate may be improved if $\mathbf{g}$ is replaced by a *full-rank* diagonal matrix where the m^{th} diagonal component is given by $(1/2)[1+\cos(\pi\, m/N)]$, $m = 0, \ldots, 2N-1$. In this case, the FDAF can be reformulated so that only three DFTs are needed overall, thereby reducing its computational complexity. Its convergence properties are similar to those of the overlap-save algorithm, particularly if the FDAF is attempting to identify an unknown globally-decaying FIR system of order $\leq N$. Another type of gradient constraint, referred to as Rosen's gradient-projection method, has been used in the context of channel equalization [2]. It is based on a constrained optimization problem which is solved by a search technique. Although the performance of this approach is similar to that of the algorithm in Fig. 5, it has a greater complexity. This FDAF was probably the first one developed based on an efficient convolution implementation.

Finally, it is possible to remove the gradient constraint entirely such that $\mathbf{G} = \mathbf{I}$. This implementation is referred to as the unconstrained frequency-domain adaptive filter [25], and it has the same configuration as in Fig. 5 except the constraint is simply bypassed so that only three DFTs are needed overall. The corresponding algorithm is

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\mu \boldsymbol{X}^H(k)\boldsymbol{E}(k) \tag{30}$$

The gradient here no longer corresponds to a linear correlation; it represents instead a circular correlation and is analogous to the circular convolution of $\mathbf{Y}(k) = \mathbf{X}(k)\,\mathbf{W}(k)$. Note, however, that the constraint on the output via (25) is still maintained by the algorithm in (30).

In general, the unconstrained FDAF does not converge to the same Wiener weight vector as that of the constrained algorithm in (24), and the corresponding steady-state MSE is greater [38], [53]. However, for the previously-mentioned FIR system identification application, the unconstrained FDAF will converge to this Wiener solution provided that the length of the unknown system is $\leq N$ [25]. In effect, the gradient constraint is being traded for a constraint on the data, as determined by the length of the unknown system [26]. Finally, although the convergence rate of the unconstrained FDAF is increased with time-varying step sizes, the misadjustment becomes greater, thus offsetting this improvement. In fact, compared to the constrained algorithm, about twice as many iterations of the unconstrained algorithm are required in order to reach the same level of misadjustment [38].

Overlap-Add Method

The overlap-add sectioning method is an alternative way of partitioning the data and reassembling the results to obtain a linear convolution [26]. The resulting FDAF algorithm [54] is essentially the same as the overlap-save FDAF in (24), except the input signal matrix is computed according to

$$\boldsymbol{X}(k) = \boldsymbol{X}'(k) + \boldsymbol{J}\boldsymbol{X}'(k-1) \tag{31}$$

where

$$\boldsymbol{X}'(k) = \mathrm{diag}\{\boldsymbol{F}[x(kN),\ldots,x(kN+N-1),\, 0,\ldots,0]^T\} \tag{32}$$

Whereas $\mathbf{X}$ (k) in (17) includes a block of the previous data samples, $\mathbf{X}'$ (k) is an *intermediate* signal matrix that contains only the current block of data augmented with N zeros. Note, however, that the sum in (31) contains the previous block of data, although it is modified by the diagonal matrix $\mathbf{J}$ which is independent of time and has elements $J_{mm} = (-1)^m$, $m = 0, \ldots, 2N-1$. The same constraint matrix, $\mathbf{g}$, is imposed on the gradient according to (23), but the overall output of the filter is computed as

$$\boldsymbol{y}(k) = \boldsymbol{k}\boldsymbol{F}^{-1}\boldsymbol{Y}(k) \tag{33}$$

This expression is similar to (25) except that

TABLE II.
FDAF ALGORITHM BASED
ON OVERLAP-ADD SECTIONING

INITIALIZATION:

$$\boldsymbol{W}(0) = [0,\ldots,0]^T$$
$$P_m(0) = \delta_m, \quad m = 0,\ldots,2N-1$$

MATRIX DEFINITIONS:

$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{I}_N & \boldsymbol{0}_N \\ \boldsymbol{0}_N & \boldsymbol{0}_N \end{bmatrix}$, $\boldsymbol{k} = [\, \boldsymbol{0}_N \;\; \boldsymbol{I}_N \,]$; sectioning constraints

$\boldsymbol{J} = \mathrm{diag}\{1, -1, 1, \ldots, -1, 1\}$; overlap–add input constraint

$\boldsymbol{F}$ = $2N \times 2N$ DFT matrix

FOR EACH NEW BLOCK OF N INPUT SAMPLES:

$$\boldsymbol{X}'(k) = \mathrm{diag}\{\boldsymbol{F}[x(kN),\ldots,x(kN+N-1),0,\ldots,0]^T\}$$

$$\boldsymbol{X}(k) = \boldsymbol{X}'(k) + \boldsymbol{J}\boldsymbol{X}'(k-1)$$

$$\boldsymbol{Y}(k) = \boldsymbol{X}(k)\boldsymbol{W}(k)$$

$$\boldsymbol{y}(k) = \boldsymbol{k}\boldsymbol{F}^{-1}\boldsymbol{Y}(k)$$

$$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{y}(k)$$

$$\boldsymbol{E}(k) = \boldsymbol{F}\boldsymbol{k}^T\boldsymbol{e}(k)$$

$$P_m(k) = \lambda P_m(k-1) + \alpha\,|X_m(k)|^2, \quad m = 0,\ldots,2N-1$$

$$\boldsymbol{\mu}(k) = \mu\ \mathrm{diag}\{P_0^{-1}(k),\ldots,P_{2N-1}^{-1}(k)\}$$

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\boldsymbol{F}\boldsymbol{g}\boldsymbol{F}^{-1}\,\mu\,(k)\boldsymbol{X}^H(k)\boldsymbol{E}(k)$$

$$\check{\mathbf{k}} = [\, \boldsymbol{I}_N \;\; \mathbf{0}_N \,] \tag{34}$$

is used instead of $\mathbf{k}$. Both the output vector, $\mathbf{Y}(k)$, and weight vector, $\mathbf{W}(k)$, are computed as before for the overlap-save FDAF. Finally, the frequency-domain error is derived by augmenting the time-domain error terms with N zeros as follows:

$$\boldsymbol{E}(k) = \boldsymbol{F}\check{\mathbf{k}}^T\boldsymbol{e}(k) \tag{35}$$

which is similar to (27) except that the zeros follow the time-domain error terms. The overall structure is summarized in Fig. 6 and the algorithm is outlined in Table II. Comparing this algorithm with the overlap-save FDAF, we see that the only difference is in the way $\mathbf{X}(k)$, $\mathbf{y}(k)$, and $\mathbf{E}(k)$ are formed. They have essentially the same computational complexity and their convergence properties are virtually identical. Although it seems that additional memory would be required here in order to store $\mathbf{X}'(k\text{-}1)$, recall that the overlap-save FDAF must store a previous block of data directly in the time domain. Thus, $2N$ samples of the input signal are always needed for the linear convolution, and the previous block of data can be stored either in the time domain (overlap-save) or in the frequency domain (overlap-add).

The original FDAF based on the overlap-add method contained a total of seven DFTs [26]. The addition described by (31) was previously done in the time domain whereby $\mathbf{X}'(k)$ and $\mathbf{X}'(k\text{-}1)$ were first transformed by IDFTs before they were added. This was required because the time-domain sequence associated with $\mathbf{X}'(k\text{-}1)$ must be circularly shifted before it is added to the time-domain version of $\mathbf{X}'(k)$. It can be shown, however, that the matrix $\mathbf{J}$ is the frequency-domain

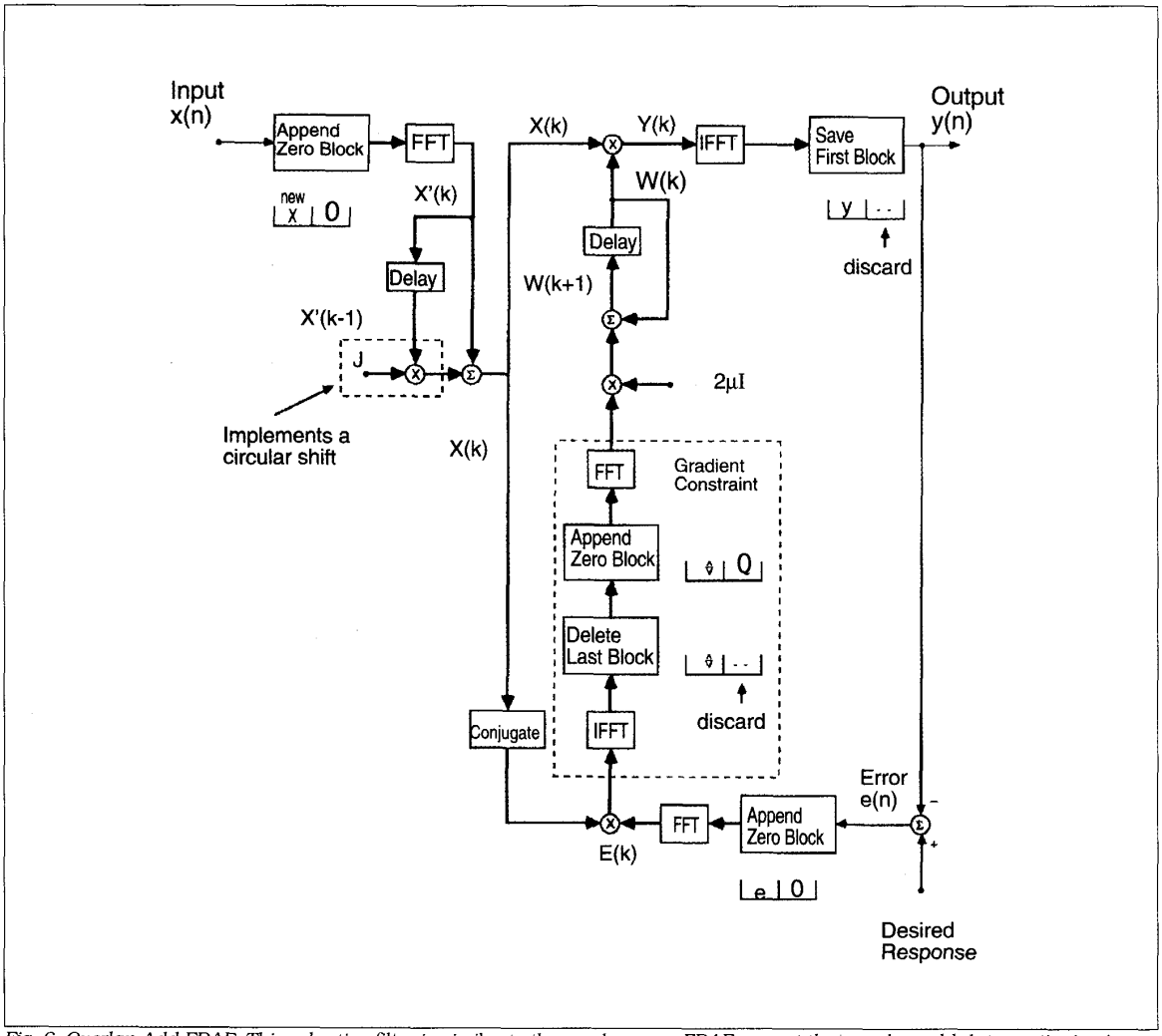


Fig. 6. Overlap-Add FDAF. This adaptive filter is similar to the overlap-save FDAF, except that overlap-add data sectioning is used to perform the linear convolution. The most recent N input samples are augmented with N zeros before the DFT is computed, and the **first** *N points of the output IDFT correspond to a linear convolution. Observe that the previous transformed input signal matrix, $\mathbf{X}'(k\text{-}1)$, must be stored for use with the current block of data, and that the sign of every other diagonal element is changed (via $\mathbf{J}$) before it is combined with $\mathbf{X}'(k)$ to generate $\mathbf{X}(k)$. The gradient constraint is identical to that of the overlap-save FDAF.*

equivalent of the circular-shift operation, thus eliminating the need for the two additional IDFTs [54].

Circular-Convolution Method

By removing the gradient constraint in Fig. 5, the $2N$-length frequency-domain weight vector no longer corresponds to N time-domain weights, thus introducing wrap-around effects that may impair the steady-state performance of the adaptive algorithm. In this section, we describe another modification to the FDAF algorithm which reduces its complexity even further, but at the expense of causing additional degradation. Recall that for the overlap-save and overlap-add FDAFs, $2N$-length DFTs were used so that N samples of the $2N$-point output vector would correspond to a linear convolution. Consider instead using only N-point DFTs which are computed once for every block of N samples (i.e., 0% overlap) as illustrated by the FDAF in Fig. 7 [21]. For this adaptive algorithm, the weight vector and the input signal matrix are given by

$$\boldsymbol{W}(k) = \boldsymbol{F}\boldsymbol{w}(k) \tag{36}$$

and

$$\boldsymbol{X}(k) = \text{diag}\{\boldsymbol{F}[x(kN),\ldots,x(kN+N-1)]^T\} \tag{37}$$

and the overall output is

$$\boldsymbol{y}(k) = \boldsymbol{F}^{-1}\boldsymbol{Y}(k) \tag{38}$$

where $\mathbf{Y}(k) = \mathbf{X}(k)\,\mathbf{W}(k)$ as before. Comparing this last expression with those of the linear-convolution FDAFs in (25) and (33), we see that the constraint matrices $\mathbf{k}$ and $\hat{\mathbf{k}}$ have been dropped, and the rank of $\mathbf{F}$ is now $M = N$. Because of these changes, it is clear that some components of $\mathbf{y}(k)$ are necessarily the result of a circular convolution (see Fig. 3).

Since the data samples are not overlapped and because the error is a linear function of the output signal and the desired response, the error is easily computed directly in the frequency domain without additional DFTs (unlike the previous FDAFs). The circular-convolution FDAF is an example of the second error configuration in Fig. 1. Taking the DFT of a block of N desired response samples, $\mathbf{D}(k) = \mathbf{F}\,\mathbf{d}(k)$, the frequency-domain error is simply

$$\boldsymbol{E}(k) = \boldsymbol{D}(k) - \boldsymbol{Y}(k) \tag{39}$$

Because there are no constraints on the gradient (i.e., $\mathbf{G} = \mathbf{I}$), the algorithm weight update has the same form as the unconstrained algorithm in (30). The complete algorithm, which was first derived by Dentino et al. [21], is summarized in Table III.

Although the circular-convolution FDAF does not require any constraints, it is still a block algorithm that has an update similar to (9). By substituting (13) into (38) and using the weight vector computed in (36), the output can be written as

$$\boldsymbol{y}(k) = \boldsymbol{F}^{-1}\boldsymbol{X}(k)\boldsymbol{F}\boldsymbol{w}(k) = \boldsymbol{\chi}(k)\boldsymbol{w}(k) \tag{40}$$

TABLE III.
FDAF ALGORITHM BASED ON CIRCULAR CONVOLUTION

INITIALIZATION:

$$\boldsymbol{W}(0) = [0,\ldots,0]^T$$

$$P_m(0) = \delta_m, \quad m = 0,\ldots,N-1$$

MATRIX DEFINITION:

$$\boldsymbol{F} = N \times N \text{ DFT matrix}$$

FOR EACH NEW BLOCK OF N INPUT SAMPLES:

$$\boldsymbol{X}(k) = \text{diag}\{\boldsymbol{F}[x(kN),\ldots,x(kN+N-1)]^T\}$$

$$\boldsymbol{D}(k) = \boldsymbol{F}\boldsymbol{d}(k)$$

$$\boldsymbol{Y}(k) = \boldsymbol{X}(k)\boldsymbol{W}(k)$$

$$\boldsymbol{E}(k) = \boldsymbol{D}(k) - \boldsymbol{Y}(k)$$

$$P_m(k) = \lambda P_m(k-1) + \alpha\,|X_m(k)|^2, \quad m = 0,\ldots,N-1$$

$$\boldsymbol{\mu}(k) = \mu\,\text{diag}\{P_0^{-1}(k),\ldots,P_{N-1}^{-1}(k)\}$$

$$\boldsymbol{W}(k+1) = \boldsymbol{W}(k) + 2\,\boldsymbol{\mu}(k)\boldsymbol{X}^H(k)\boldsymbol{E}(k)$$

Since $\mathbf{X}(k)$ is a diagonal matrix, $\boldsymbol{\chi}(k) = \boldsymbol{F}^{-1}\boldsymbol{X}(k)\boldsymbol{F}$ is a *circulant* matrix[8] [36], [27] such that each row and column uniquely define the entire matrix. The first column of $\boldsymbol{\chi}(k)$ contains N samples of the input block $\{x(kN), \ldots, x(kN+N-1)\}$. Taking the IDFT of this FDAF algorithm (i.e., (30)) and using the definition of $\boldsymbol{\chi}(k)$ yields

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + 2\mu\boldsymbol{\chi}^{\mathrm{T}}(k)\boldsymbol{e}(k) \tag{41}$$

The block gradient associated with (41) is

$$\hat{\nabla}(k) = \sum_{m=0}^{N-1}\boldsymbol{\chi}_m(k)\,e(kN+m) \tag{42}$$

where $\boldsymbol{\chi}_m^T(k)$ is the m^{th} row[9] of $\boldsymbol{\chi}(k)$. Thus, (42) has a block update which is similar to the linear correlation in (9), except the error terms here are being correlated with a *circularly-shifted* version of the input vector. Similarly, the output vector in (40) is the result of a circular convolution between the time-domain weights and the input signal.

[8] A circulant matrix is a square matrix with rows determined by successive right circular ("wrap-around") shifts of the first row.

[9] In this paper, the rows (and columns) of a matrix are numbered beginning with zero (e.g., row m with $m = 0, \ldots, N-1$).

The obvious advantage of using only N-point DFTs is that they have less computational complexity. This result, coupled with the savings obtained by removing the gradient constraint, leads to a significant reduction in the overall complexity of the FDAF. The main disadvantage is that this FDAF often has a degraded performance because it is only an approximate implementation of the block adaptive algorithm in (9). Its convergence properties [55], [56], [27] are generally quite different from those of the previous FDAFs because of the distorting effects of the circular convolution.

The adaptive weights in (41) converge to the Wiener weight vector for this filter, which is quite different from the one that minimizes the block MSE. In effect, each weight in (30) is adjusted to minimize the MSE associated with its own frequency bin rather than the global MSE corresponding to the overall output of the filter. If the bins were less correlated, then minimizing these two performance functions would lead to similar results. However, there is considerable spectral overlap, so we cannot expect (41) to have a steady-state performance similar to that of the linear-convolution FDAFs. One possible exception may occur, for example, in an adaptive line enhancer (ALE) application if the sinusoids are widely separated such that they lie essentially in different frequency bins.

Computational Complexity

The computational complexities of the block convolution-based FDAFs are now summarized for com-

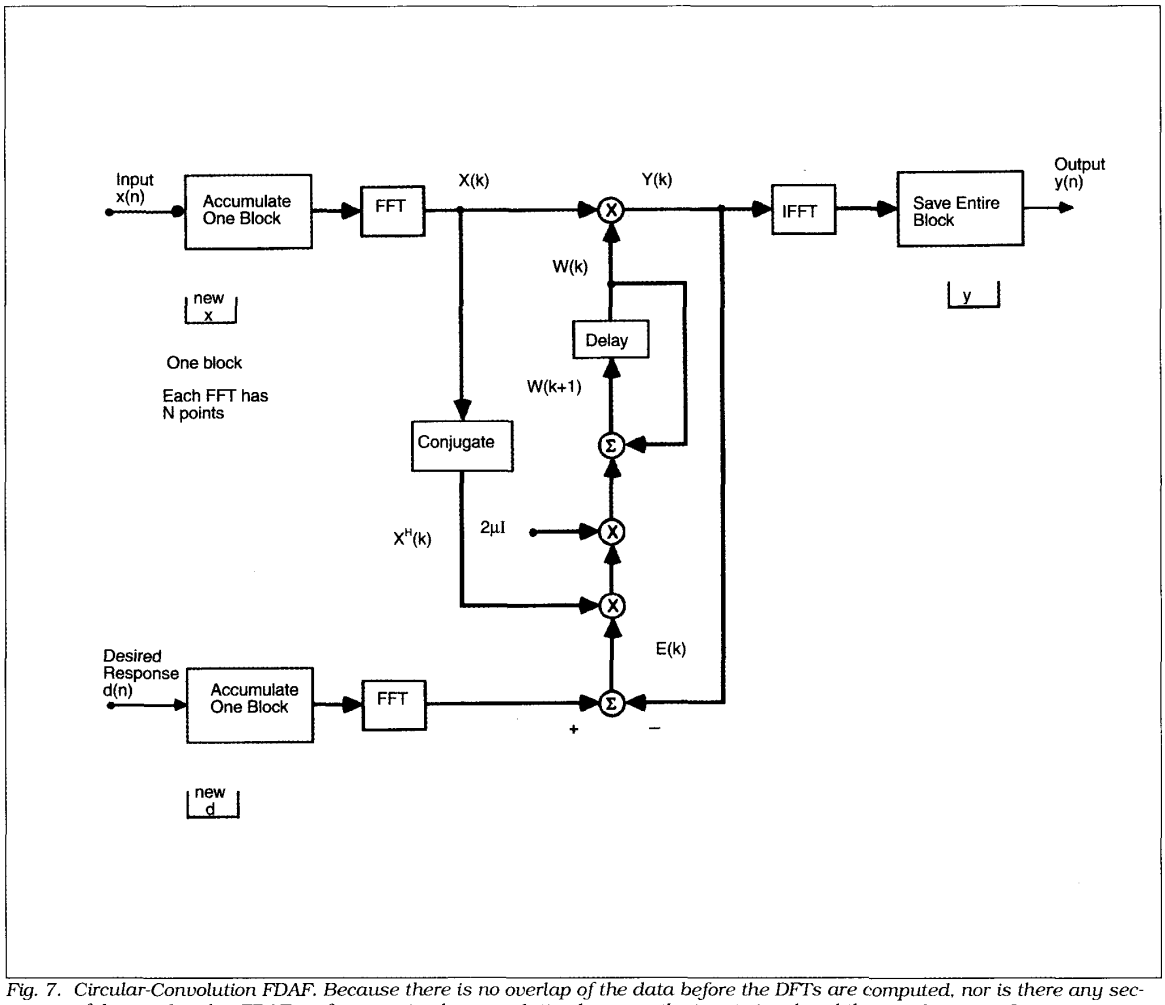


Fig. 7. Circular-Convolution FDAF. Because there is no overlap of the data before the DFTs are computed, nor is there any sectioning of the results, this FDAF performs a circular convolution between the input signal and the weight vector. In contrast to the previous FDAFs, the error terms here are computed directly in the frequency domain, although it is straightforward to perform the equivalent operation in the time domain. A gradient constraint is not used because there are only N frequency-domain weights, and these are related to the corresponding time-domain weights via an N-point IDFT. From an implementation point of view, the DFTs are computed only after N data samples have been accumulated. It is useful, however, to imagine that the DFTs are computed for every new data sample, and that the output signals are instead downsampled by a factor of N (maximally decimated). In this way, the DFT operates as a bank of bandpass filters, and this FDAF can be viewed as a special case of a subband adaptive filter. Although the circular-convolution FDAF was originally designed such that the DFTs are computed once for each block of data (0% overlap), they could instead be computed more frequently in order to reduce the aliasing distortion.

parison with the nonblock time-domain LMS algorithm. Although we examine only the total number of multiplications for each implementation, these results provide reasonably accurate comparative estimates of their overall complexity. In an actual implementation, several other issues would have to be considered, such as the number of additions, storage requirements, system transport delays, hardware designs, etc.

The nonblock time-domain LMS algorithm with N real weights requires N multiplications to compute its output and another N multiplications to update the weight vector, for a total of $2N$ real multiplications to produce each output sample. Thus, $2N^2$ real multiplications are required for every N output samples. The complexity of each FDAF will be compared to this amount in the form of a *ratio*, as summarized in Table IV for several values of N [27].

Each M-point FFT (and IFFT) requires approximately $M \log_2(M)$ real multiplications [57] (assuming real-valued input data and radix-2 FFT algorithms) where M is either N or $2N$ depending on whether the convolution is circular or linear, respectively. The frequency-domain output vector requires $4M$ real multiplications as does the gradient correlation, so the total complexity is $PM \log_2(M) + 8M$ real multiplications where P is the number of DFTs. The complexity ratios were thus computed using

$$\text{Complexity ratio} = (PM\log_2(M) + 8M)/2N^2 \quad (43)$$

For the linear-convolution overlap-save FDAF ($P = 5$ and $M = 2N$), the total number of real multiplications is $10N \log_2(2N) + 16N$. Since the unconstrained FDAF requires only three FFTs, its complexity is $6N \log_2(2N) + 16N$ real multiplications. Finally, for the circular-convolution FDAF where $M = N$ and $P = 3$, its overall complexity is $3N \log_2(N) + 8N$ real multiplications.

By examining Table IV, we see that a substantial reduction in the computational complexity can be expected when using any of the block frequency-domain implementations. As expected, the circular-convolution FDAF has the least complexity while the linear-convolution FDAF with the gradient constraint has the greatest. Each of the block FDAFs has a savings when the filter length is relatively small (≈ 64), and they have considerable complexity advantages when N exceeds 1000. This dramatic reduction in the number of multiplications is obtained because of the FFT algorithms and the block updating.

Sliding DFT and Frequency-Sampling Methods

In addition to the block frequency-domain techniques, *nonblock* FDAFs have been developed that exploit the orthogonality properties of the DFT to improve the convergence rate of the LMS algorithm. Figure 8 shows a filter configuration where a single DFT is used to transform the input signal vector into an equivalent frequency-domain representation. The weight vector and input signal matrix are the same as those defined in (36) and (37) for the circular-convolution FDAF. However, unlike the previous methods, the DFT here is computed for *each* new input sample, and the algorithm is updated at the incoming data rate (i.e., $k = n$). When operated in this manner, the DFT is often referred to as a *sliding* DFT [51]. The output vector $\mathbf{Y}(n)$ is still computed according to (13), and its terms are simply added

TABLE V.
FDAF ALGORITHM BASED ON THE SLIDING DFT

INITIALIZATION:

$$\mathbf{W}(0) = [0,\ldots,0]^T$$

$$P_m(0) = \delta_m, \quad m = 0,\ldots,N-1$$

MATRIX DEFINITION:

$$\mathbf{F} = N \times N \text{ DFT matrix}$$

FOR EACH NEW INPUT SAMPLE:

$$\mathbf{X}(n) = \text{diag}\{\mathbf{F}[x(n),\ldots,x(n-N+1)]^T\}$$

$$\mathbf{Y}(n) = \mathbf{X}(n)\mathbf{W}(n)$$

$$e(n) = d(n) - \mathbf{1}^T\mathbf{Y}(n)$$

$$\mathbf{E}(n) = \mathbf{1}e(n)$$

$$P_m(n) = \lambda P_m(n-1) + \alpha |X_m(n)|^2, \quad m = 0,\ldots,N-1$$

$$\boldsymbol{\mu}(n) = \mu \, \text{diag}\{P_0^{-1}(n),\ldots,P_{N-1}^{-1}(n)\}$$

$$\mathbf{W}(n+1) = \mathbf{W}(n) + 2\,\boldsymbol{\mu}(n)\mathbf{X}^H(n)\mathbf{E}(n)$$

TABLE IV.
FDAF COMPUTATIONAL COMPLEXITY RATIOS

FDAF ALGORITHM	FILTER SIZE N						
	16	32	64	128	256	1024	2048
Linear Convolution	2.063	1.188	0.672	0.375	0.207	0.062	0.033
Unconstrained Gradient	1.438	0.813	0.453	0.250	0.137	0.040	0.021
Circular Convolution	0.625	0.359	0.203	0.113	0.062	0.019	0.010

together to yield the time-domain output. As a result, the output error is

$$e(n) = d(n) - \mathbf{1}^T\mathbf{Y}(n) \tag{44}$$

where $\mathbf{1}$ is a column vector of ones. Alternatively, if a sliding IDFT of $\mathbf{Y}(n)$ was computed, then only the first element of the resulting output vector would be needed because the first component of an IDFT is the sum of the input vector elements.[10]

The adaptive algorithm is thus given by

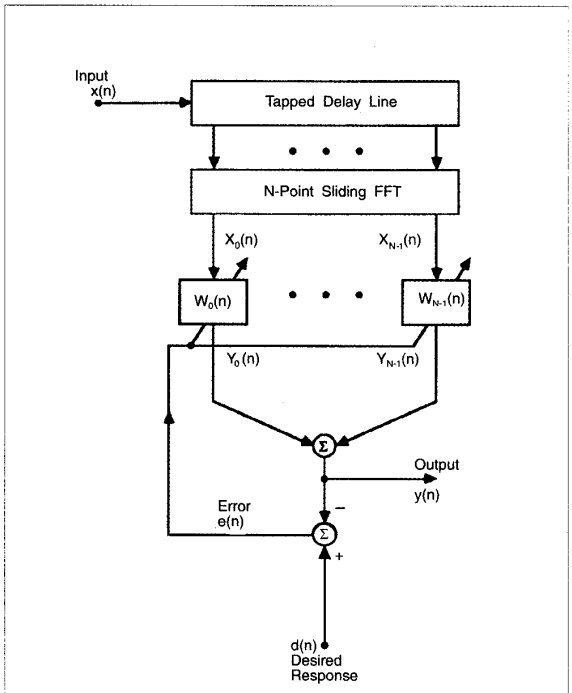
$$\mathbf{W}(n+1) = \mathbf{W}(n) + 2\boldsymbol{\mu}(k)\mathbf{X}^H(n)\mathbf{1}e(n) \tag{45}$$

There is no gradient constraint ($\mathbf{G} = \mathbf{I}$) and the error vector is $\mathbf{E}(n) = \mathbf{1}\,e(n)$. As shown in Fig. 8, each weight is updated with the same error term $e(n) = d(n) - y(n)$. It is important that this algorithm employ a set of time-varying step sizes, which we emphasize by including $\boldsymbol{\mu}(k)$ in (45). It therefore becomes possible to obtain a more uniform convergence rate [58], [59] for some input signals because of the orthogonality properties of the DFT. This is essentially the only advantage of the sliding DFT FDAF; it does not have the computational efficiency of the previous block FDAFs. Other transformations can also be used, such as the discrete cosine transform (DCT), and these might be more appropriate for certain types of signals (such as speech). This FDAF is often referred to as the transform-domain[11] LMS algorithm, first developed by Narayan et al. [43]. The complete algorithm is summarized in Table V.

The Karhunen-Lóeve transform (KLT) [39] would be the ideal transform because it takes into account the eigenvalues of the input signal correlation matrix. However, since this matrix is assumed to be unknown or may even be time-varying (because of nonstationary signals), the KLT must be approximated by a time-invariant transform such as the DCT. The FDAF in Fig. 8 is an attempt to compensate for the eigenvalue disparity by using a fixed transform and individual adaptive step sizes for each of the weights, with a modest increase in

[10] Actually, the first component of the IDFT would be $\mathbf{1}^T\mathbf{Y}(n)$ divided by N because of the way we have defined $\mathbf{F}^{-1}$.

[11] Transform domain as applied only to the FDAF in Fig. 8 is a somewhat misleading description because all of the adaptive algorithms considered in this paper operate in the "transform domain."



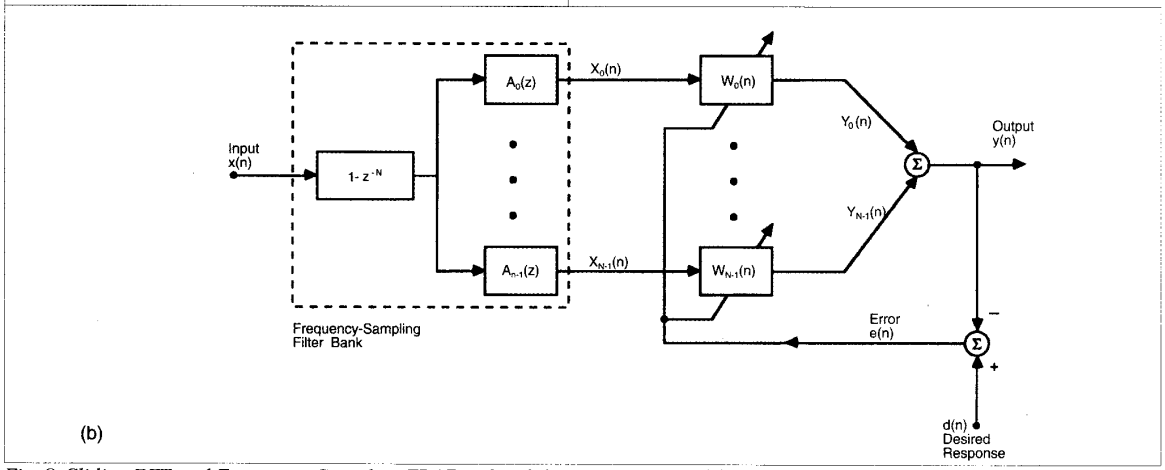


Fig. 8. Sliding DFT and Frequency-Sampling FDAFs. The sliding DFT FDAF in (a) is not a block adaptive filter because the DFT is computed for each new input sample. This configuration is similar to a tapped-delay-line LMS adaptive filter, except that a nonadaptive transformation (the DFT) is first performed on the input sequence. This allows each adaptive weight to have its own time-varying step size with a value that changes according to the signal power in that frequency bin. Because the DFT functions as a bank of narrow-band bandpass filters, it is possible to replace it by a frequency-sampling filter bank as shown in (b), which is a more efficient implementation.

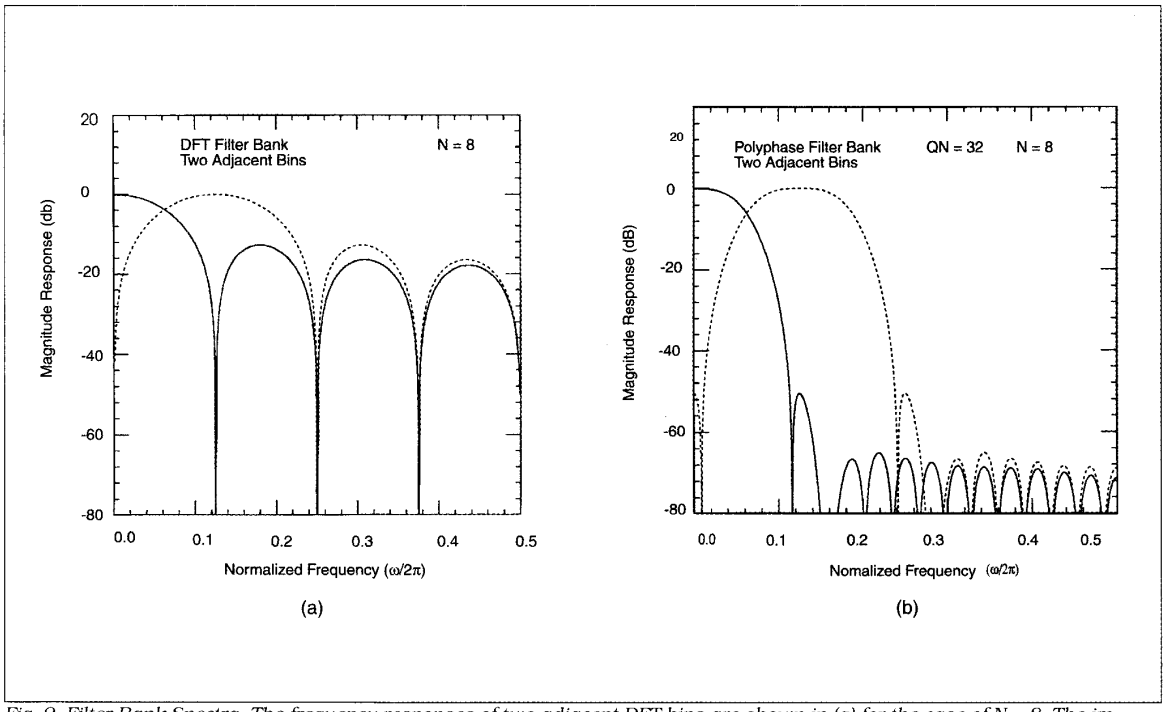


Fig. 9. Filter Bank Spectra. The frequency responses of two adjacent DFT bins are shown in (a) for the case of N = 8. The impulse response of the m^{th} DFT bin is given by

$$h_m(n) = e^{-j2\pi\, m\, n/N}, \quad n = 0,\ldots,N-1,$$

which is the m^{th} row of $\mathbf{F}$. *Clearly, there is a large degree of spectral overlap, indicating that the DFT output signals are only approximately uncorrelated. Observe that the first sidelobe is only about 13 dB down from the main lobe. This suggests the possibility of using other filter bank structures that have less spectral overlap. The frequency responses of two adjacent bins of a polyphase filter bank with N = 8 bands are shown in (b). The impulse response of the prototype filter has QN = 32 coefficients, and these are given by [46]*

$$h(n) = \frac{N \sin[(\pi/N)(n - QN/2)]}{\pi(n - QN/2)} g(n), \quad n = 0,\ldots,QN-1$$

where g (n) is a Hamming window. Observe that the first sidelobe is now about 50 dB down from the main lobe, indicating that the output signals of this filter bank are much less correlated.

the computational complexity. The RLS algorithm also partially compensates for this eigenvalue spread using, in effect, an estimate of the input signal correlation matrix. The RLS algorithm has a greater complexity than the standard LMS algorithm, but it more closely approximates the advantages of the ideal KLT-based algorithm.

The DFT in Fig. 8 can be realized more efficiently with a frequency-sampling (FS) structure[12] [51]. When the DFT is computed for each input sample, the transfer function from the input $x(n)$ to the m^{th} DFT output $X_m(n)$ is

$$H_m(z) = \frac{X_m(z)}{X(z)} = \frac{1 - z^{-N}}{1 - e^{-j2\pi\, m/N} z^{-1}} \qquad (46)$$

Consequently, the implied FIR filter bank of the DFT can be replaced with this more efficient IIR configuration. The numerator of (46) represents a set of zeros that are equally spaced around the unit circle,[13] and the denominator contains a single pole that exactly cancels one of these zeros. The FDAF based on this frequency-sampling structure is also shown in Fig. 8 where we see that the DFT operates as a bank of narrow-band bandpass filters. The frequency responses for the filters associated with two adjacent DFT bins are shown in Fig. 9a, demonstrating that they have a large degree of spectral overlap. Alternative forms of this FS adaptive filter have been described in [44], [60], [61].

Because the pole for each of the FS filters lies exactly on the unit circle, this realization is actually marginally stable so that any round-off errors could result in filter instability [62]. As a result, the poles and zeros can be

[12] This implementation of the DFT is also called a recursive DFT [60].

[13] The zeros of (46) are referred to as the N roots of unity and they are given by $z_m = e^{j2\pi\, m/N}$, $m = 0, \ldots, N-1$.

NOMENCLATURE

ALE	Adaptive line enhancer.
BLMS	Block least-mean square.
CMA	Constant modulus algorithm.
DCT	Discrete cosine transform.
DFT	Discrete Fourier transform.
FDAF	Frequency-domain adaptive filter.
FIR	Finite impulse response.
FFT	Fast Fourier transform.
FM	Frequency modulation.
FS	Frequency sampling.
IDFT	Inverse discrete Fourier transform.
IFFT	Inverse fast Fourier transform.
IIR	Infinite impulse response.
KLT	Karhunen-Lóeve transform.
LMS	Least-mean square.
MSE	Mean-square error.
QMF	Quadrature mirror filter.
RLS	Recursive least squares.
L	Decimation factor and block size ($1 \le L \le N$).
M	Size of the DFT (N or $2N$).
N	Number of time-domain filter coefficients.
NS	Number of data samples.
QN	Length of the filter bank prototype filter.
$\mathbf{F}$	DFT matrix (rank N or $2N$).
$\mathbf{I}$	Identity matrix.
$\mathbf{0}$	Zero matrix.
$\mathbf{1}$	Vector with all elements = 1.
μ	Algorithm step size.
λ	Forgetting factor of the power estimate ($\lambda = 1 - \alpha$).
ω	Radian frequency.
diag[·]	Operator that forms a diagonal matrix.
E[·]	Expectation operator.

moved slightly inside the unit circle by replacing z^{-1} in (46) with $z^{-1}\beta$ where $0 < \beta < 1$ [63]. For example, with $\beta = 0.99$ the poles and zeros all lie on a circle with radius 0.99 so that even without exact pole-zero cancellation, the FS filters will be stable.

This filter bank representation suggests that it may be possible to use other filter designs which have less spectral overlap (see Fig. 9b), such that the filter bank output signals may be decimated. Because of the large spectral overlap of the FS implementation, any decimation will cause severe aliasing distortion. By minimizing this overlap, the effects of aliasing can be reduced and decimation will lead to a considerable reduction in complexity, while still taking advantage of the orthogonality properties of the filter bank to improve the convergence rate. Of course, once the signals are decimated, it will be necessary to have a synthesis filter bank which reconstructs the output signal of the adaptive filter. This approach is referred to as *subband* adaptive filtering [29].

SUBBAND IMPLEMENTATIONS

Filter Banks

As an alternative to the convolution-based frequency-domain adaptive filters, subband techniques can also achieve computational efficiency by decimating the signals before the adaptive processing. One motivation for this approach is provided by the circular-convolution FDAF where the DFT can be viewed as a filter bank whose output is maximally decimated by the factor N. As mentioned above in connection with the frequency-sampling structure, the DFT filter bank has a large degree of spectral overlap which can lead to severe aliasing distortion. Thus, it would be desirable to use alternative filter bank designs that have a reduced spectral overlap without a large additional complexity.

The theory of multirate signal processing for non-adaptive filters is well established, and there are many techniques for designing the filter banks [31], [64]-[66]. The idea of subband filtering evolved primarily from work on speech and image processing where it has been found that by splitting signals into smaller frequency bands, a considerable reduction in the coding complexity is possible [67]. The same ideas have been extended to other signal processing applications, including adaptive filtering. The subband adaptive filters are shorter in length than the full-band adaptive filter, although the total number of coefficients is usually the same. The complexity advantage is thus achieved by the downsampling process.

A subband filter is comprised of an analysis filter bank which "splits" the input signal into narrow frequency bins so that the subband signals can be decimated with minimal aliasing distortion. After transmission, for example, these subband signals can be upsampled (interpolated) and then processed by a synthesis filter bank to generate an estimate of the original signal. The synthesis bank has a form that is similar to the analysis bank, and it is possible to design them such that their combination results in "perfect" reconstruction of the original signal. Perfect reconstruction [64] is a property of filter banks whereby the magnitude and phase of the original signal is completely restored, except for a possible end-to-end delay.

The quadrature mirror filter (QMF) bank is probably the most well-known subband implementation [66]. It is comprised of simple filter components, and the perfect reconstruction property is achieved by designing them to satisfy a certain structural relationship. These filter components are usually a frequency-shifted version of a prototype low-pass filter which has the desired frequency cutoff (transition-band) specifications. The complexity of the QMF realization increases with the number of subbands as well as the sharpness of the transition band. Several implementations are possible, including tree structures and polyphase realizations [31].

Subband Adaptive Filtering

When designing the analysis filter bank, it is desirable to have approximately nonoverlapping frequency bins so that the subband adaptive filters operate independently. However, practical filters have a finite transition band so that a filter design with nonoverlapping bins often has spectral "holes" that can adversely affect the steady-state performance of an adaptive algorithm. For example, if the adaptive filter is configured as an ALE, it is possible for the unknown sinusoids to be located in these spectral gaps, and thus they may not be "observed" by the adaptive process. On the other hand, a large degree of spectral overlap causes aliasing distortion as well as a reduced convergence rate. Obviously, there is a trade-off when designing the filter banks, so the specific configuration will depend on the requirements of the application. One of the first subband adaptive filters was based on a transmultiplexer[14] design [14], [27] which included some spectral overlap. In order to reduce the aliasing, the filter bank output signals are often decimated by a factor less than the number of bands (e.g., $L = N / 2$). Additional examples of this are presented in [27], [11], [12].

The basic configuration of a subband adaptive filter is shown in Fig. 10, which is similar to the circular-convolution FDAF except that the DFTs are replaced with more general filter banks, and the downsampling is performed as part of the implementation.[15] The adaptive algorithm has the same form as the unconstrained recursion in (30) where $\mathbf{X}(k)$ and $\mathbf{E}(k)$ are derived as shown in Fig. 10. A recent subband adaptive filter design had nonoverlapping frequency bins [10], but the convergence properties of the algorithm were generally not acceptable for the reasons discussed above. An improved design with overlapping bins was then developed, but it was necessary to include adaptive *cross-terms* to reduce the aliasing distortion because the filter bank output signals were maximally decimated ("critically sampled") [9], [29].

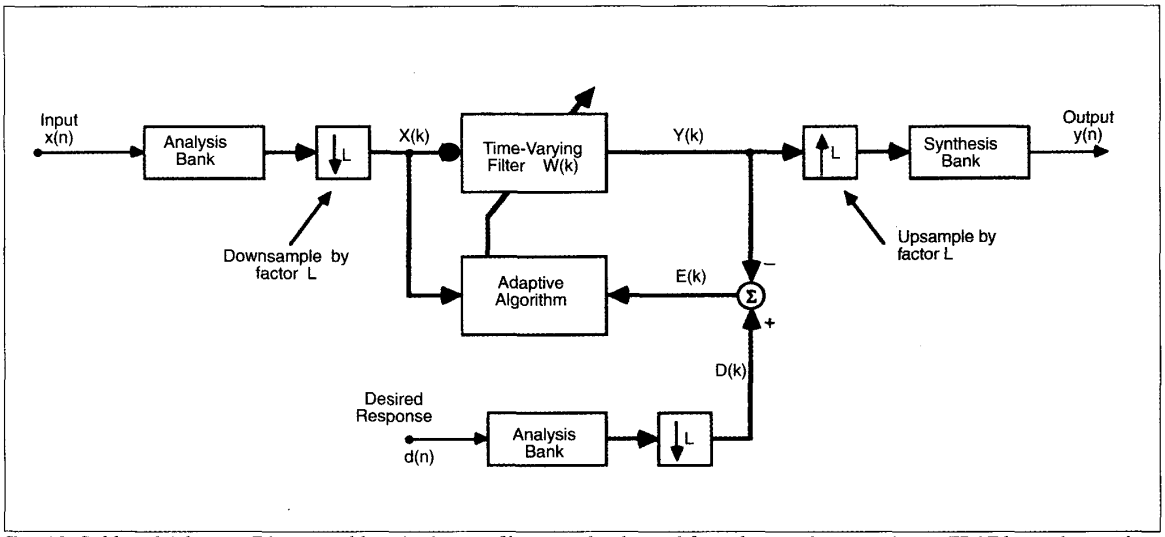


Fig. 10. Subband Adaptive Filter. A subband adaptive filter may be derived from the circular-convolution FDAF by replacing the DFTs with a set of filter banks that have less spectral overlap. The circular-convolution FDAF is essentially maximally decimated (by N), and this causes severe aliasing distortion. The transformed signals here are usually downsampled by a factor L < N, thereby minimizing the effects of aliasing. Several types of filter banks can be used, such as those based on polyphase designs.

Consider the two-band case (without cross-terms) shown in Fig. 11 that has analysis filters $A_0(z)$ and $A_1(z)$. Usually, one chooses the analysis filters such that $A_1(z) = A_0(-z) = A(z)$ where $A(z)$ represents the *prototype* filter. The synthesis filters are then chosen as $B_0(z) = A_0(z)$ and $B_1(z) = A_1(z)$. As a result, the intermediate "frequency-domain" output signals can be written as

$$\begin{bmatrix} Y_0(z) \\ Y_1(z) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} W_0(k, z) & 0 \\ 0 & W_1(k, z) \end{bmatrix}\begin{bmatrix} A(z^{1/2}) & A(-z^{1/2}) \\ A(-z^{1/2}) & A(z^{1/2}) \end{bmatrix}\begin{bmatrix} X(z^{1/2}) \\ X(-z^{1/2}) \end{bmatrix} \tag{47}$$

where $X(z)$ is the input and $\{W_m(k, z)\}$ represent the adaptive subfilters (their time-varying nature is emphasized by the block time argument k). The superscripts of z are a standard notation used to describe multirate systems [64]. Equation (47) can be rewritten in the following compact form with the obvious correspondences between the terms:

[14] Transmultiplexer is a somewhat confusing name for a polyphase implementation since this term is used to describe a device in telecommunication systems that converts between time-division multiplexing and frequency-division multiplexing [68].

[15] Recall that the circular-convolution FDAF doesn't explicitly downsample the DFT output signals; instead the DFTs are computed only once for each block of data.

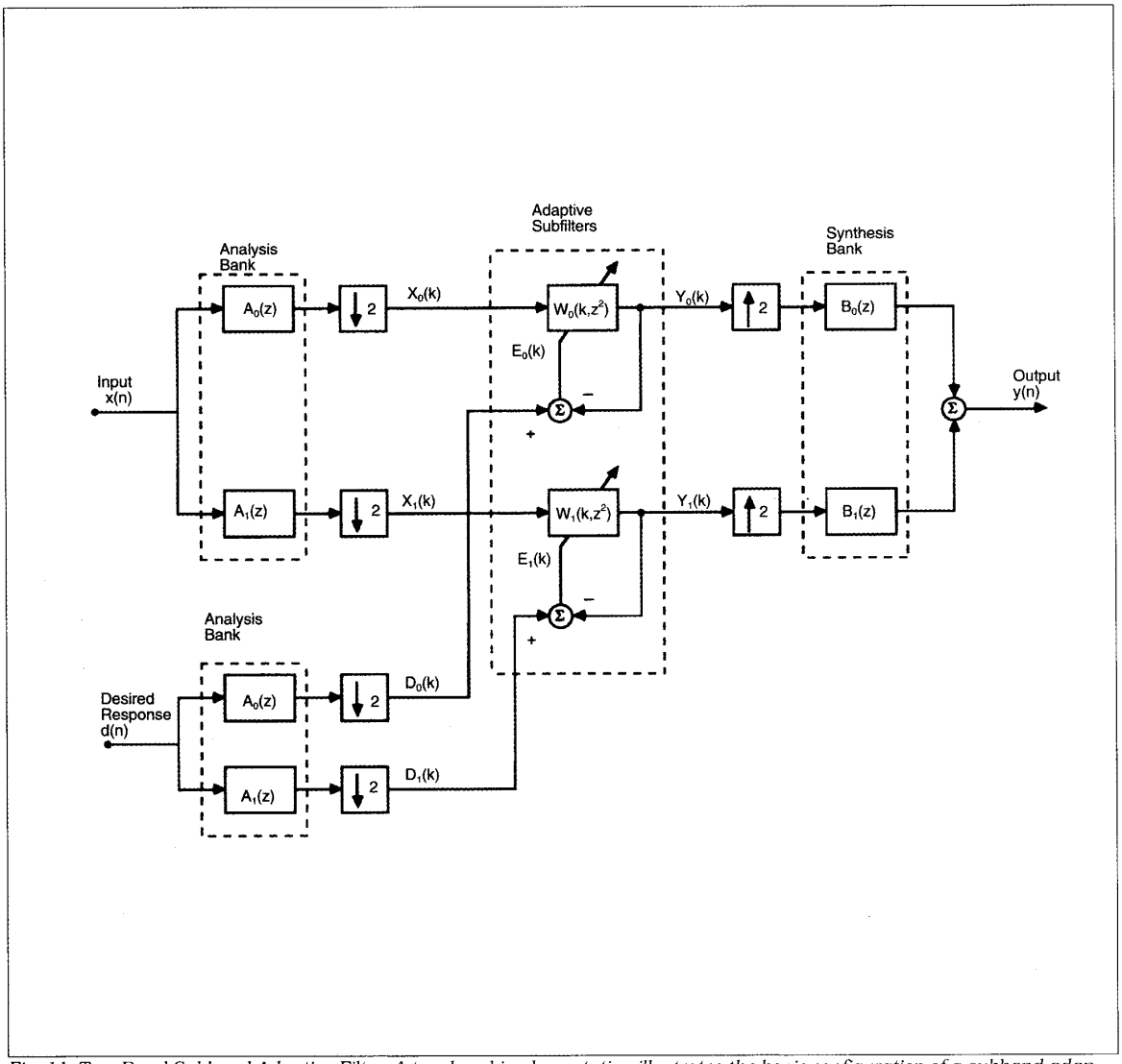


Fig. 11. Two-Band Subband Adaptive Filter. A two-band implementation illustrates the basic configuration of a subband adaptive filter, and it is easily generalized to an arbitrary number of bands. In this case, the two filters of the QMF bank correspond to low-pass and high-pass designs such that their combined output contains the entire spectrum of the transformed signal. The filter bank signals are downsampled by a factor of 2, processed by independent adaptive filters, upsampled back to the original rate, and then recombined by the reconstruction (synthesis) filter bank. The adaptive filters usually have several coefficients, although the number is often decreased as the number of subbands increases so that the total number of coefficients matches that of the corresponding time-domain adaptive filter.

$$\mathbf{Y}(z) = \frac{1}{2}\mathbf{W}(k, z)\mathbf{A}(z^{1/2})\mathbf{X}(z^{1/2}) \tag{48}$$

Note that $\mathbf{W}(k, z)$ is diagonal by design. However, if we consider an application where the adaptive filter is configured to identify an unknown transfer function $C(z)$ (e.g., in an echo-cancellation application), then it can be shown that when the subband filters are time-invariant [29],

$$\mathbf{W}(\mathbf{z}) \approx cz^{p} \cdot$$

$$\begin{bmatrix} A^2(z^{1/2})C(z^{1/2}) - A^2(-z^{1/2})C(-z^{1/2}) & A(z^{1/2})A(-z^{1/2})[C(-z^{1/2}) - C(z^{1/2})] \\ A(z^{1/2})A(-z^{1/2})[C(z^{1/2}) - C(-z^{1/2})] & A^2(z^{1/2})C(-z^{1/2}) - A^2(-z^{1/2})C(z^{1/2}) \end{bmatrix} \tag{49}$$

where c is a constant scalar and p is a positive integer. This expression is an approximation to the Wiener solution and thus represents the "optimal" structure of the adaptive filter. Clearly, this is not a diagonal matrix so that the adaptive matrix in (48) should be replaced by the following more general form:

$$\mathbf{W}(k, z) = \begin{bmatrix} W_{00}(k, z) & W_{01}(k, z) \\ W_{10}(k, z) & W_{11}(k, z) \end{bmatrix} \tag{50}$$

Although this modified subband adaptive filter has a reduced aliasing distortion, the convergence rate is somewhat degraded because the subbands are no longer uncoupled. The computational complexity is also slightly increased. For the general M-band case, it is not necessary that all adaptive subfilters be cross-coupled

because only adjacent bins are strongly overlapping, so the additional complexity due to the cross-terms increases only linearly with the number of subbands.

There have been several other subband implementations for adaptive filtering. One approach uses QMF banks with nonoverlapping subbands to avoid aliasing distortion, and it includes an *auxiliary* adaptive filter (possibly with a multi-band frequency response) that compensates for the spectral holes [69], [6]. The length of this additional filter can usually be very short (compared to the main subbands), so the additional complexity is less than that introduced by the cross-terms above. The auxiliary filter can also be downsampled, resulting in even less additional complexity. Although there have been a few other designs (see, e.g., [5], [30]), subband adaptive filtering is still largely an open area of research. Their convergence properties have not been analyzed and it is difficult to predict their behavior in a general way.

IIR ALGORITHMS AND NONLINEAR ERROR FUNCTIONS

IIR Implementations

Compared to FIR filter implementations, there has been relatively little work on frequency-domain realizations for adaptive IIR filtering. An important advantage with IIR structures is that they can provide improved steady-state performance with fewer coefficients because of the feedback, thereby further reducing the overall complexity [70], [17], [18]. One of the first attempts involved a frequency-domain model of a time-domain ALE that was used to study its convergence properties [71]. This representation has a specific structure that was designed as part of the model and, as such, it is not considered to be a practical implementation. Recently, frequency-domain IIR realizations have been developed for general-purpose signal processing applications, including both equation-error and output-error formulations [19].

The equation-error formulation is similar to adaptive FIR filtering because the mean-square-equation error is a quadratic function of the weights; in addition to the feedforward coefficients that weight the input signal, there are "feedback" coefficients that similarly weight the *delayed* desired response. A frequency-domain implementation [72] of this formulation is shown in Fig. 12, which is similar to the linear convolution FDAF in Fig. 5 based on the overlap-save data sectioning method. Note that the desired response is also processed by a DFT, weighted by a set of coefficients, and the result is then added to the output of the feedforward component of the filter. Because of the "feedback" coefficients, a total of eight DFTs are now required because of the three DFTs which process the desired response. This adaptive filter has convergence properties similar to that of the overlap-save FIR FDAF, but the converged coefficients may be biased depending on the statistics of the input signal and the desired response [19].

An output-error FDAF for adaptive IIR filtering can be derived from the nonblock sliding DFT adaptive filter in Fig. 8 by replacing the weights with a set of single-pole, single-zero adaptive filters [73]. This configuration is actually a parallel-form implementation whereby the DFT preprocesses the input signal before adaptation. Other types of filter banks can be used and, depending on their spectral overlap, they are capable of improving the convergence properties of the adaptive filter. A frequency-sampling implementation of the DFT can also be employed here. Adaptive IIR filters based on the output-error formulation may converge to local minima [17], which is a potential drawback similar to the bias problem of the above equation-error formulation.

Nonlinear Error Functions

This paper has focused on the LMS algorithm which has a linear error function, i.e., $e(n) = d(n) - y(n)$. As such, the two error formulations described in Fig. 1 are often equivalent and it is usually straightforward to develop FDAF algorithms where the error is computed in either the time or frequency domains. On the other hand, when the error is a nonlinear function of the data, it is not always clear how to derive the equivalent frequency-domain error. Even if this were possible, the complexity would very likely be increased or the frequency-domain error may only be an approximation.

For example, consider the constant modulus algorithm (CMA) [74], [34] which has the following time-domain weight update:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - 4\mu \boldsymbol{x}^{*}(n) y(n) [\,|y(n)|^2 - r^2] \qquad (51)$$

where r is a predetermined constant and the superscript * denotes complex conjugate. CMA is a "blind" adaptive algorithm [75] since it does not require an explicit desired response signal, and its goal is to force the modulus of the equalizer output, $y(n)$, to be a constant value. This property is useful when a constant-modulus signal (e.g., FM) is transmitted across a noisy channel that has linear and nonlinear distortion; by restoring the modulus of the received signal, it may be possible to sufficiently reduce the bit error rate in order to switch the equalizer to a decision-directed mode. There are four versions[16] of CMA; the algorithm in (51) is generally referred to as CMA 2-2.

Clearly, the "error" term of this algorithm, given by $-2y(n)[\,|y(n)|^2 - r^2\,]$, is a nonlinear function of the filter output. In this case, it is more appropriate to compute the error in the time domain. It is not immediately obvious what the corresponding frequency-domain error formulation is, and it is unlikely that any approximations will produce satisfactory performance. Another example of a nonlinear error computation is the decision-directed equalization scheme whereby the demodulator decisions are used as the "desired response" signal [40].

[16] The performance function of CMA p-q is $\xi_{pq} = E[\,|\,|y(n)|^p - r^p|^q\,]$ where p and q are integers chosen to be either 1 or 2.

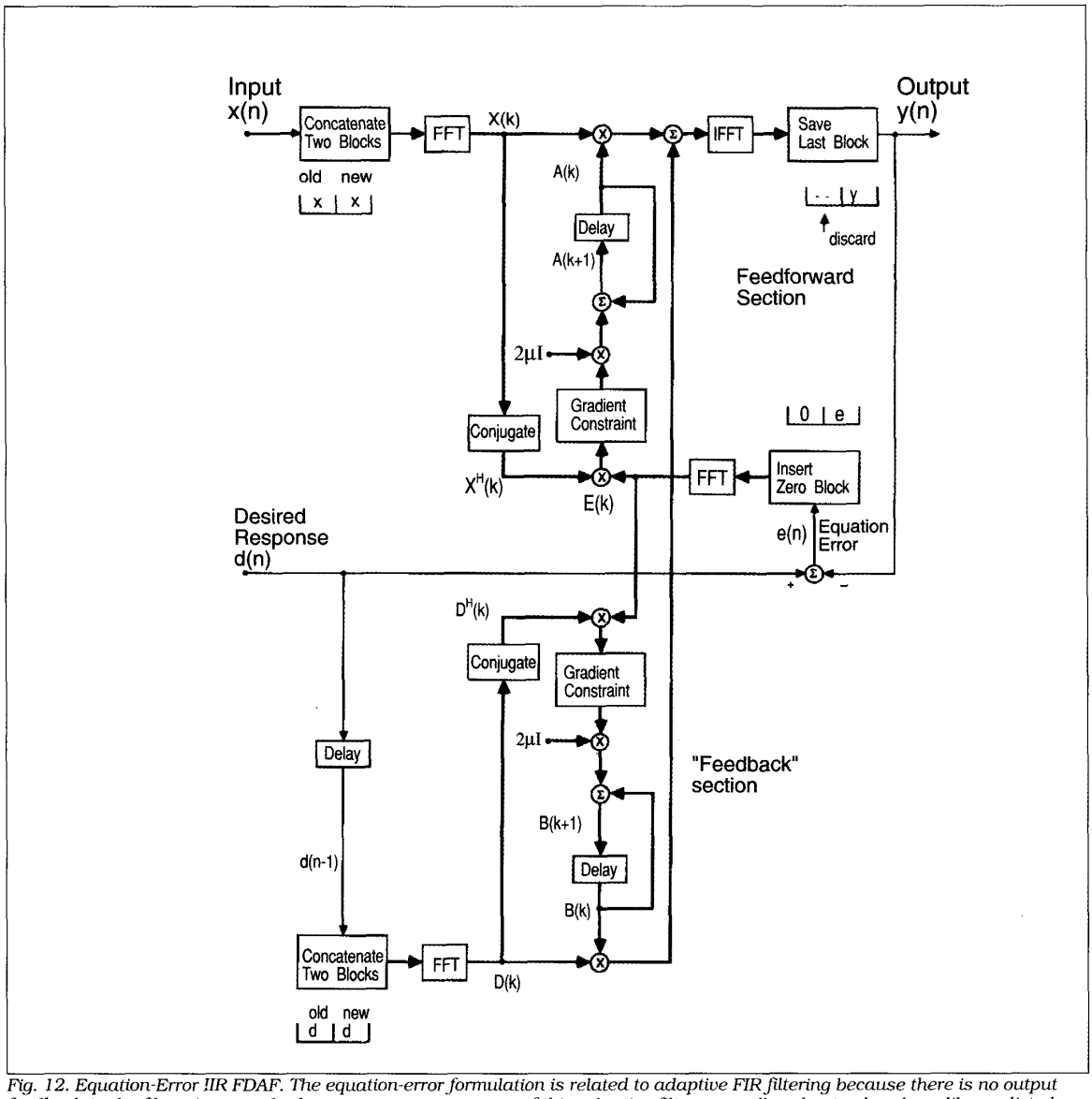


Fig. 12. Equation-Error IIR FDAF. The equation-error formulation is related to adaptive FIR filtering because there is no output feedback in the filter. As a result, the convergence properties of this adaptive filter are well understood and readily predicted. The main difference is that the desired response is filtered by a set of "feedback" coefficients, which is analogous to the set of feedforward coefficients that weight the input signal. All of the previous FDAF configurations can be modified to accommodate the equation-error formulation, as demonstrated here for the overlap-save linear convolution method.

Figure 13 shows a subband implementation of CMA using polyphase filter banks [76]. Because this filter bank has an inherent group delay (transport delay), the error signal $\mathbf{E}(k)$ is not a function of the current input $\mathbf{X}(k)$ [77]. As a result, it is necessary to *delay* the transformed input as $\mathbf{X}(k - \Delta)$ before computing the gradient correlation, otherwise the performance of the algorithm will be severly deteriorated. The amount of delay depends on the type of filter banks used, and it is usually needed only when the error is computed in the time domain. In general, $\Delta = \lfloor (QN-1)/L$ where QN is the length of the prototype filter, N is the number of adaptive weights, and L is the block size (decimation factor) [76]. The operator $\lfloor u$ retains the greatest integer $\leq u$. For the DFT filter bank in Fig. 9a (with maximum decimation), the delay is always $\Delta = 0$ for any value of N, whereas for a polyphase filter bank based on the prototype filter in Fig. 9b with $QN = 32$ coefficients and $L = N = 8$, the delay is $\Delta = 3$. Thus, the input signal matrix, $\mathbf{X}(k)$, of the linear-convolution FDAFs (overlap-save and overlap-add) does not require a block delay even though the error is computed in the time domain. The delay is required only when the length of the filter bank prototype filter relative to the block size exceeds the above threshold.

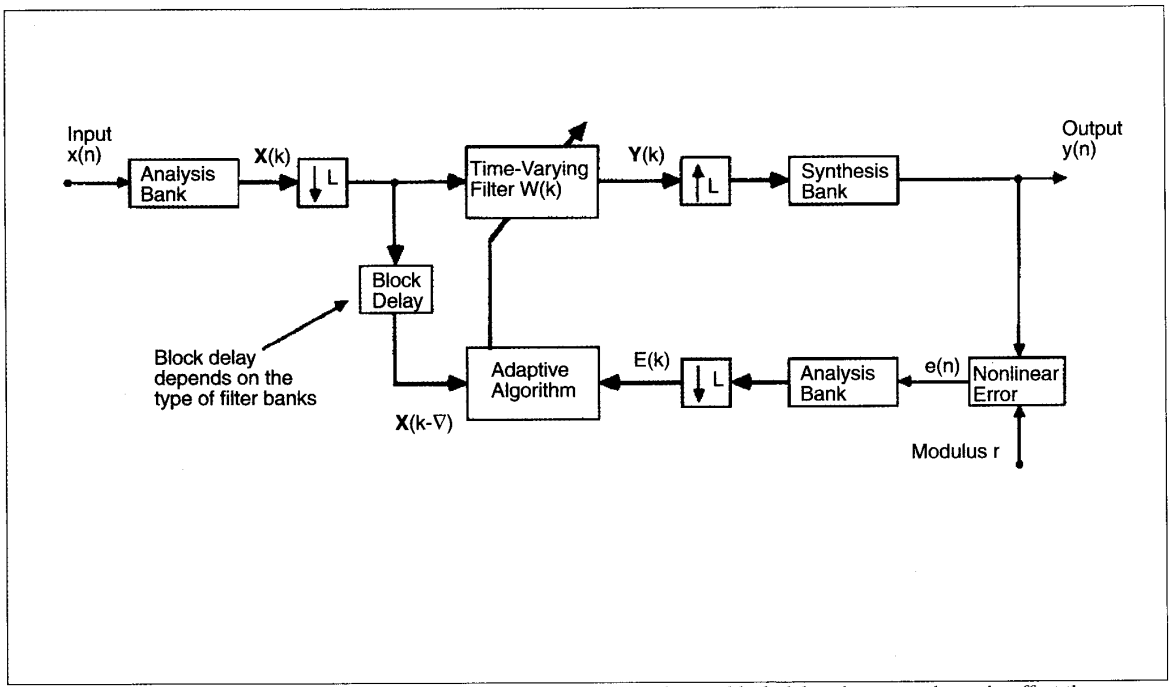


Fig. 13. Block Delay. Filter bank implementations usually introduce an inherent block delay that can adversely affect the convergence properties of the adaptive filter unless the algorithm signals are properly matched in time. This potential problem often arises with nonlinear error formulations where the error signal must be computed in the time domain. As shown in the figure, the transformed input signal must be delayed before it is cross-correlated with the transformed error vector. The amount of delay depends on the number of subbands and the decimation factor, as well as the type of filter banks.

CONCLUSION AND SUMMARY

An overview of several frequency-domain adaptive filters (FDAFs) has been presented. Generally, these algorithms can be divided into two classes depending on the type of frequency-domain transformation. One class is based on the DFT and its ability to generate either a circular convolution or linear convolution, as determined by the scheme used to partition the data. The other class is based on subband filtering techniques where the signals are processed by a set of filter banks. Both approaches are considered to be multirate adaptive filters since the adaptive processing is performed at a lower sampling rate than that of the incoming data, thus reducing the computational complexity. These parallel configurations often result in faster convergence rates than their time-domain counterparts and they are more amenable to hardware implementations. On the other hand, they introduce an end-to-end delay that could be a problem in applications such as telecommunications. In addition, the weights are kept fixed while a block of data is accumulated, which is not desirable for tracking purposes when the data are highly nonstationary. Nevertheless, the computational and convergence rate advantages of frequency-domain adaptive filters can be considerable, and it is expected that they will become more widely used in many signal processing applications.

ACKNOWLEDGMENTS

This work was supported by Applied Signal Technology, Inc., Rockwell International Corporation, and the University of California MICRO Program. The author greatly appreciates the valuable comments and suggestions of Greg Clark, Rich Gooch, Mariane Petraglia, and Piet Sommen.

John J. Shynk (S'78, M'86, SM'91) was born in Lynn, MA, on June 20, 1956. He received the B.S. degree in systems engineering from Boston University, Boston, MA, in 1979; the M.S. degree in electrical engineering and in statistics, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1980, 1985, and 1987, respectively.

From 1979 to 1982, he was a Member of Technical Staff in the Data Communications Performance Group at AT&T Bell Laboratories, Holmdel, NJ, where he formulated performance models for voiceband data communications. He was a Research Assistant from 1982 to 1986 in the Electrical Engineering Department at Stanford University where he worked on frequency-domain implementations of adaptive IIR filter algorithms. From 1985 to 1986, he was also an instructor at Stanford, teaching courses on digital signal processing and adaptive systems. Since 1987, he has been an

Assistant Professor in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara. He is also Associate Director of the Center for Information Processing Research. His current research interests include developing and analyzing efficient adaptive signal processing algorithms for applications in blind equalization and neural networks. He is an Associate Editor for adaptive filtering of the *IEEE Transactions on Signal Processing* and is Technical Program Chairman of the 1992 International Joint Conference on Neural Networks in Baltimore. He was also Co-Chairman of the 1990 IEEE Communication Theory Workshop. Dr. Shynk is a member of Tau Beta Pi, Sigma Xi, and INNS.

REFERENCES

[1] Special Session on Frequency Domain Adaptive Filtering, in *Proc. 23rd Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1989, pp. 663-698.

[2] T. Walzman and M. Schwartz, "Automatic equalization using the discrete frequency domain," *IEEE Trans. Inform. Theory*, vol. IT-19, no. 1, pp. 59-68, Jan. 1973.

[3] G. Picchi and G. Prati, "Self-orthogonalizing adaptive equalization in the discrete frequency domain," *IEEE Trans. Commun.*, vol. COM-32, no. 4, pp. 371-379, Apr. 1984.

[4] M. J. Ready, S. H. Goldberg, and R. P. Gooch, "Architecture considerations for frequency domain adaptive equalizers," in *Proc. 23rd Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1989, pp. 687-691.

[5] S. C. Pohlig, "A filter bank structure for adaptive nulling," in *Proc. 21st Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1987, pp. 458-462.

[6] V. S. Somayazulu, S. K. Mitra, and J. J. Shynk, "Adaptive line enhancement using multirate techniques," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Glasgow, Scotland, May 1989, pp. 928-931.

[7] W. Kellermann, "Kompensation akustischer echos in frequenzteilbändern," *Frequenz*, vol. 39, no. 7/8, pp. 209-215, July/Aug. 1985.

[8] M. R. Asharif, F. Amano, S. Unagami, and K. Murano, "Acoustic echo canceler based on frequency bin adaptive filter (FAF)," in *Proc. IEEE Int. Conf. Global Telecommun.*, Tokyo, Japan, Nov. 1987, 1940-1944.

[9] A Gilloire, "Experiments with subband acoustic echo cancellers for teleconferencing," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Dallas, TX, Apr. 1987, pp. 2141-2144.

[10] H. Yasukawa, S. Shimada, and I. Furukawa, "Acoustic echo canceller with high speech quality," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Dallas, TX, Apr. 1987, pp. 2125-2128.

[11] W. Kellermann, "Analysis and design of multirate systems for cancellation of acoustical echoes," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, New York, NY, Apr. 1988, pp. 2570-2573.

[12] E. Hänsler, "Adaptive echo compensation applied to the hands-free telephone problem," in *Proc. IEEE Int. Symp. Circuits Systems*, New Orleans, LA, May 1990, pp. 279-282.

[13] M. Dentino, "Frequency domain adaptive correlator," in *Proc. 11th Asilomar Conf., Circuits, Systems, Computers*, Pacific Grove, CA, Nov. 1977, pp. 267-271.

[14] G. C. Copeland, "Transmultiplexers used as adaptive frequency sampling filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Paris, France, May 1982, pp. 319-322.

[15] C. F. N. Cowan and P. M. Grant, Eds., *Adaptive Filters*, Englewood Cliffs, NJ: Prentice-Hall, 1985.

[16] F. A. Reed, P. L. Feintuch, and N. J. Bershad, "The application of the frequency domain LMS adaptive filter to split array bearing estimation with a sinusoidal signal," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-33, no. 1, pp. 61-69, Feb. 1985.

[17] C. R. Johnson, Jr., "Adaptive IIR filtering: Current results and open issues," *IEEE Trans. Inform. Theory*, vol. IT-30, no. 2, pp. 237-250, Mar. 1984.

[18] J. R. Treichler, "Adaptive algorithms for infinite impulse response filters," in *Adaptive Filters*, C. F. N. Cowen and P. M. Grant, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985, ch. 4, pp. 60-90.

[19] J. J. Shynk, "Adaptive IIR filtering," *IEEE ASSP Magazine*, vol. 6, no. 2, pp. 4-21, Apr. 1989.

[20] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms: Theory and Implementation*. New York: Wiley, 1985.

[21] M. Dentino, J. M. McCool, and B. Widrow, "Adaptive filtering in the frequency domain," *Proc. IEEE*, vol. 66, no. 12, pp. 1658-1659, Dec. 1978.

[22] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block adaptive filtering," in *Proc. IEEE Int. Symp. Circuits Systems*, Houston, TX, Apr. 1980, pp. 384-387.

[23] E. R. Ferrara, Jr., "Fast implementation of LMS adaptive filters," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-28, no. 4, pp. 474-475, Aug. 1980.

[24] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Circuits Systems*, vol. CAS-28, no. 6, pp. 584-592, June 1981.

[25] D. Mansour and A. H. Gray, Jr., "Unconstrained frequency-domain adaptive filter," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-30, no. 5, pp. 726-734, Oct. 1982.

[26] G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time- and frequency-domain realization of FIR adaptive digital filters," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-31, no. 5, pp. 1073-1083, Oct. 1983.

[27] E. R. Ferrara, Jr., "Frequency-domain adaptive filtering," in *Adaptive Filters*, C. F. N. Cowen and P. M. Grant, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985, ch. 6, pp. 145-179.

[28] P. C. W. Sommen, "Partitioned frequency domain adaptive filters," in *Proc. 23rd Asil. Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1989, pp. 677-681.

[29] A. Gilloire and M. Vetterli, "Adaptive filtering in subbands," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, New York, NY, Apr. 1988, pp. 1572-1575.

[30] S. K. Mitra, M. R. Petraglia, and A. Mahalanobis, "Structural subband implementation of adaptive filters," in *Proc. 24th Asil. Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1990, pp. 232-236.

[31] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[32] P. C. W. Sommen, private communication, Oct. 1990.

[33] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[34] J. R. Treichler and B. G. Agee, "A new approach to multipath correction of constant modulus signals," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-31, no. 2, pp. 459-472, Apr. 1983.

[35] D. F. Elliott and K. R. Rao, *Fast Transforms: Algorithms, Analyses, Applications*. Orlando, FL: Academic, 1982.

[36] R. M. Gray, "On the asymptotic eigenvalue distribution of Toeplitz matrices," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 6, pp. 725-730, Nov. 1972.

[37] M. J. Shensa, "The spectral dynamics of evolving LMS adaptive filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Washington, DC, Apr. 1979, pp. 950-953.

[38] P. C. W. Sommen, P. J. Van Gerwen, H. J. Kotmans, and A. J. E. M. Janssen, "Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function," *IEEE Trans. Circuits Systems*, vol. CAS-34, no. 7, pp. 788-798, July 1987.

[39] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, Second Ed., 1991.

[40] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, Second Ed., 1989.

[41] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," in *IRE WESCON Conv. Rec.*, Pt. 4, 1960, pp. 96-104.
[42] B. Widrow and J. M. McCool, "A comparison of adaptive algorithms based on the methods of steepest descent and random search," *IEEE Trans. Antennas and Propagation*, vol. AP-24, no. 5, pp. 615-637, Sept. 1976.
[43] S. S. Narayan and A. M. Peterson, "Frequency domain least-mean-square algorithm," *Proc. IEEE*, vol. 69, no. 1, pp. 124-126, Jan. 1981.
[44] R. R. Bitmead and B. D. O. Anderson, "Adaptive frequency sampling filters," *IEEE Trans. Circuits Systems*, vol. CAS-28, no. 6, pp. 524-534, June 1981.
[45] A. Feuer, "Performance analysis of the block least mean square algorithm," *IEEE Trans. Circuits Syst.*, vol. CAS-32, no. 9, pp. 960-963, Sept. 1985.
[46] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[47] L. Pelkowitz, "Frequency domain analysis of wrap-around error in fast convolution algorithms," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-29, no. 3, pp. 413-422, June 1981.
[48] B. Widrow, J. M. McCool, and M. Ball, "The complex LMS algorithm," *Proc. IEEE*, vol. 63, no. 4, pp. 719-720, Apr. 1975.
[49] L. J. Griffiths, "An adaptive lattice structure for noise-cancelling applications," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Tulsa, OK, Apr. 1978, pp. 87-90.
[50] R. R. Bitmead and B. D. O. Anderson, "Performance of adaptive estimation algorithms in dependent random environments," *IEEE Trans. Automat. Control*, vol. AC-25, no. 4, pp. 788-794, Aug. 1980.
[51] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.
[52] G. A. Clark, S. R. Parker, and S. K. Mitra, "Efficient realization of adaptive digital filters in the time and frequency domains," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Paris, France, May 1982, pp. 1345-1348.
[53] J. C. Lee and C. K. Un, "Performance analysis of frequency-domain block LMS adaptive digital filters," *IEEE Trans. Circuits Systems*, vol. 36, no. 2, pp. 173-189, Feb. 1989.
[54] P. C. W. Sommen and J. A. K. S. Jayasinghe, "On frequency-domain adaptive filters using the overlap-add method," in *Proc. IEEE Intl. Symp. Circuits Systems*, Espoo, Finland, June 1988, pp. 27-30.
[55] N. J. Bershad and P. L. Feintuch, "Analysis of the frequency domain adaptive filter," *Proc. IEEE*, vol. 67, no. 12, pp. 1658-1659, Dec. 1979.
[56] F. A. Reed and P. L. Feintuch, "A comparison of LMS adaptive cancellers implemented in the frequency domain and the time domain," *IEEE Trans. Circuits Systems*, vol. CAS-28, no. 6, pp. 610-615, June 1981.
[57] N. K. Jablon, "Complexity of frequency-domain adaptive filtering for data modems," in *Proc. 23rd Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1989, pp. 692-698.
[58] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-31, no. 3, pp. 609-615, June 1983.
[59] J. C. Lee and C. K. Un, "Performance of transform-domain LMS adaptive digital filters," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-34, no. 3, pp. 499-510, June 1986.
[60] G. A. Clark, M. A. Soderstrand, and T. G. Johnson, "Transform domain adaptive filtering using a recursive DFT," in *Proc. IEEE Int. Symp. Circuits Systems*, Kyoto, Japan, June 1985, pp. 1113-1116.
[61] M. R. Petraglia and S. K. Mitra, "On the use of filter banks in adaptive filtering," in *Proc. 22nd Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Oct. 1988, pp. 30-34.
[62] J. M. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. Circuits Systems*, vol. CAS-34, no. 7, pp. 821-833, July 1987.
[63] J. J. Shynk and B. Widrow, "Bandpass adaptive pole-zero filtering," in *Proc. IEEE Intl. Conf. Acoust., Speech, Sig. Proc.*, Tokyo, Japan, Apr. 1986, pp. 2107-2110.
[64] P. P. Vaidyanathan, "Quadrature mirror filter banks, M-band extensions and perfect-reconstruction techniques," *IEEE ASSP Magazine*, vol. 4, no. 3, pp. 4-20, July 1987.
[65] M. Vetterli, "A theory of multirate filter banks," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. 35, no. 3, pp. 356-372, Mar. 1987.
[66] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proc. IEEE*, vol. 78, no. 1, pp. 56-93, Jan. 1990.
[67] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[68] J. Bellamy, *Digital Telephony*. New York: Wiley, Second. Ed., 1991.
[69] S. K. Mitra, H. Babic, and V. S. Somayazulu, "A modified perfect reconstruction QMF bank with an auxiliary channel," in *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, May 1989, pp. 2132-2135.
[70] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.
[71] N. J. Bershad and P. L. Feintuch, "The recursive adaptive LMS filter - A line enhancer application and analytical model for the mean weight behavior," *IEEE Trans. Acoust., Speech, Sig. Proc.*, vol. ASSP-28, no. 6, pp. 652-660, Dec. 1980.
[72] W. B. Mikhael, A. S. Spanias, and F. H. Wu, "Fast frequency domain implementation of a block IIR filter with applications," in *Proc. IEEE Intl. Symp. Circuits Syst.*, Espoo, Finland, June 1988, pp. 285-288.
[73] J. J. Shynk and R. P. Gooch, "Frequency-domain adaptive pole-zero filtering," *Proc. IEEE*, vol. 73, no. 10, pp. 1526-1528, Oct. 1985.
[74] D. N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. Commun.*, vol. COM-28, no. 11, pp. 1867-1875, Nov. 1980.
[75] A. Benveniste and M. Goursat, "Blind Equalizers," *IEEE Trans. Commun.*, vol. COM-32, no. 8, pp. 871-883, Aug. 1984.
[76] C. K. Chan, M. R. Petraglia, and J. J. Shynk, "Frequency-domain implementations of the constant modulus algorithm," in *Proc. 23rd Asilomar Conf. Signals, Systems, Computers*, Pacific Grove, CA, Nov. 1989, pp. 663-669.
[77] J. R. Treichler, S. L. Wood, and M. G. Larimore, "Convergence rate limitations in certain frequency-domain adaptive filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Proc.*, Glasgow, Scotland, May 1989, pp. 960-963.